البرمجة باستخدام *Visual Basic.NET 2008*

وسام الدين محمد

اتفاقية الاستخدام

هذا الكتاب وقف لله عز وجل، يخضع لجميع قواعد الوقف الإسلامي، وهذا يعني أنه يجوز لكل مسلم ومسلمه إعادة توزيعه في صورته الإلكترونية أو إعادة طبعه بشرط عدم التربح منه أو تغيير شئ من محتوياته. وقد جعلته هكذا أبتغاء مرضاة ربي وآمل في أن يحط عني شئ من أوزاري، فلا تنسني من صالح دعائك.

تقديم

هذا الكتيب في الأصل قد أعد ليكون مدخلاً للبرمجة العامة باستخدام Visual Basic لطلاب دبلوم نظم المعلومات الجغرافية في الأكاديمية العربية للعلوم والتكنولوجيا بالاسكندرية. وقد مر بمراحل شتى حيث كتبته لأول مرة ليغطي Visual Basic 6 ومن ثم أضطررت لإعادة كتابة أقسام كثيرة منه عند صدور Visual Basic.Net 2003 ثم تم أضافة بعد التعديلات لتوافق الإصدارة Visual Basic.Net 2005. وبالرغم من توقفي عن التدريس إلا أنني رأيت أن أجدد هذا الكتيب ليكون متوافقاً مع الإصدار Visual Basic.NET 2008 خاصة مع المظاهر الجديدة الكثيرة التي أضفيت على لغة البرمجة الأساسية في هذا الأصدار، وقد أضطرتني هذه المظاهر الجديدة إلى إضافة القسم رقم ٥ لعرض تقنية Windows Presentation Foundation WPF الجديدة، وتعديل الأقسام رقم ٨ و ٩ و ١٠. وكنت قد أضفت جزء عن تقنية LINQ ثم قمت بحذفه بناء على نصيحة عدد من الزملاء العاملين في الحقل حيث أن هذه التقنية غير مستقرة بعد.

صمم هذا الكتاب ليكون منهجاً تعليمياً لطلاب متخصصين في نظم المعلومات بصورة عامة ونظم المعلومات الجغرافية بصورة خاصة، ومن ثم فإن الموضوعات التي يضمها هي تلك الموضوعات الأساسية الشائعة التي يحتاجها مبرمج نظم المعلومات، ولا يتطرق إلى الموضوعات المتقدمة مثل برمجة الشبكات أو برمجة الهواتف المحمولة مثلا.

وقد أضفت عدد من التدريبات إلى كل قسم من أقسام هذا الكتاب بغرض أن يلم القارئ بالموضوع الذي يقدمه القسم من خلال التطبيق. وتعمدت أن لا تعتمد هذه التدريبات على أي مواد خارجية إلا في الأحوال التي تتطلب هذا حتى لا يتشتت القارئ بين موضوع الكتاب وغيره من الموضوعات. كما تعمدت أن تكون معظم هذه التدريبات مبنية على بعضها البعض حتى يدرك القارئ الطريقة التي تتطور بها التطبيقات.

وفي الختام أتمنى أن يكون هذا الكتيب نافعاً لطلاب والدراسين، وأن يجد طريقة إلى مكتبة وعقل كل طالب أو باحث عربي في هذا الحقل.

جدول المحتويات

جدول الأشكال

مفاهيم أساسية

1.1مقدمة

في عام ١٩٦٣ أبتكر جون كيميني John Kemeny وتوماس كورتز Thomas Kurtz لغة Basic وهي الكلمة التي تختصر العبارة " Beginners All Purposes Symbolic Instruction Code" أو "شفرة التعليمات الرمزية متعددة الأغراض للمبتدئين". وكما هو واضح من تسميتها فإنها صممت كي تحقق احتياجات البرمجة للمبرمجين غير المحترفين من الهواة والطلاب والباحثين غير المتخصصين في علوم الحاسوب. وقد لاقت لغة Basic نجاحاً واسعاً على النطاق الذي صممت من أجله.

وفي مايو ١٩٩١ طرحت شركة Microsoft نسخة مطورة من Basic منحتها اسم Visual Basic حيث استخدمت اللفظة المضافة Visual للتعبير عن الوظيفة الجديدة التي أضيفت لهذه اللغة كأول لغة مصممة لتطوير تطبيقات ذات واجهات استخدام رسومية Graphical User Interface تعمل على نظام التشغيل Operating System الثوري الذي كانت تروج Microsoft له منذ عام ١٩٨٥ وهو Microsoft Windows.

ومنذ ذلك التاريخ لعبت Visual Basic دور اللغة الأولى لتطوير تطبيقات Microsoft Windows، وقد طرحت Microsoft منها تسعة إصدارات، ومنذ الإصدارة السابعة التي أطلق عليها اسم Visual Basic 2003، أعيد تصميم Visual Basic بحيث أصبحت تعمل وفقاً للتقنية الجديدة لتطبيقات Microsoft المسماة .NET Framework ومن ثم حملت الاسم الجديد Visual Basic.NET حيث أصبحت أبسط وأكثر فاعلية وقابلية في تطوير التطبيقات التي تتماشى مع الاتجاهات المعاصرة في الحوسبة مثل خدمات الوب وتطبيقات الحواسيب الكفية. وتعد الإصدارة الأحدث اليوم هي Visual Basic.NET 2008 التي طرحت في أول يناير ٢٠٠٨.

في هذا القسم سنتعرض لعدد من المفاهيم الأساسية اللازمة للتعرف على ما هي Visual Basic.NET 2008 وكيف تعمل.

1.2لغة واحدة ونسخ مختلفة

عندما تم تصميم Visual Basic للمرة الأولى، راعى مصمميها أن تكون سهلة الاستخدام وفي نفس الوقت قادرة على الاستجابة لمتطلبات مختلف المبرمجين، المبتدئين منهم والمحترفين على قدر سواء. وهذا ما جعل إمكانات Visual Basic تزداد إصداره تلو أخرى، وهذا ما نتج عنه وضع أصبح فيه كثير من المبرمجين ذات الحاجات المحدودة يحتاجون أن يتعلمون تقنيات لن يحتاجونها أبداً في تطبيقاتهم، كما كان عليهم أن يدفعوا ثمن هذه التقنيات عندما يشترون أدوات تطوير Visual Basic.

ولتفادي هذا الوضع، قامت Microsoft بطرح عدة نسخ من Visual Basic تتوافق مع حاجات المبرمجين المختلفة. وقد ضم الإصدار Visual Basic.NET 2008 ثلاثة نسخ رئيسية هي: النسخة المخففة Express Edition، والنسخة القياسية Standard Edition، والنسخة الاحترافية Professional Edition.

النسخة المخففة Visual Basic 2008 Express هي نسخة تضم التقنيات الأساسية التي يحتاجها المبرمجين المبتدئين والهواة لتطوير تطبيقات قائمة بذاتها Stand Alone Applications تعمل تحت نظام التشغيل Microsoft Windows، وهي مصممة لكي تخدم الأغراض التعليمية غير الربحية ومن ثم فإن الحصول عليها يمكن عن طريق تنزيلها من موقع Microsoft على الانترنت، حيث يتم تنزيلها منفردة – بدون أي من تقنيات Visual Studio.NET 2008 الأخرى – ومجاناً.

النسخة Visual Basic.NET 2008 Standard Edition هي نسخة مخصصة يمكن استخدامها لتطوير تطبيقات مختلفة سواء تلك القائمة بذاتها أو تطبيقات الوب Web Applications أو تطبيقات الخادم/العميل Server/Client Applications أو تطبيقات النظام Windows CE المخصص للحواسيب الكفية[1] Windows CE Applications. ويمكن الحصول على هذه النسخة منفردة أو ضمن مجموعة تقنيات Visual Studion.NET 2008 مقابل دفع ثمنها.

النسخة الاحترافية Visual Basic.NET 2008 Professional Edition هي نسخة يمكن استخدامها لتطوير جميع أنواع تطبيقات الحواسيب والوب، وتتميز هذه النسخة بصورة خاصة أنه يمكن استخدامها في تطوير التطبيقات التي يعمل عليها أكثر من مبرمج

[1] والهواتف النقالة Mobile Phones وأنظمة الملاحة Navigation Systems بالسيارات والمركبات البحرية والجوية وأجهزة نظام الموقع العالمي Global Positioning System GPS.

في آن واحد، و هو ما لا توفره النسخة القياسية، وأيضاً يمكن الحصول على هذه النسخة منفردة أو ضمن مجموعة تقنيات Visual Studion.NET 2008 مقابل دفع ثمنها.

وقبل الانتهاء من هذا القسم لابد أن نقدم نبذة عن Visual Studio.NET 2008. هذا اللفظ Visual Basic.NET 2008 تقصد به Microsoft حزمة من أدوات تطوير مختلف التطبيقات تضم ثلاثة لغات برمجة بصورة أساسية هي Visual Basic.NET 2008 و Visual C++ 2008 و Visual C#.NET 2008، تعمل هذه اللغات الثلاثة من خلال بيئة تطوير متكاملة Integrated Development Environment IDE بحيث يسهل على المبرمج أن يقوم بتطوير حلول متعددة اللغات، أضف على ذلك إلى أن هذه الحزمة تعتمد على تقنية .NET Framework والتي توفر مجموعة من التقنيات الأساسية لتطوير التطبيقات المختلفة وتطبيقات الـ ASP وتطبيقات خدمات الوب بصورة خاصة.

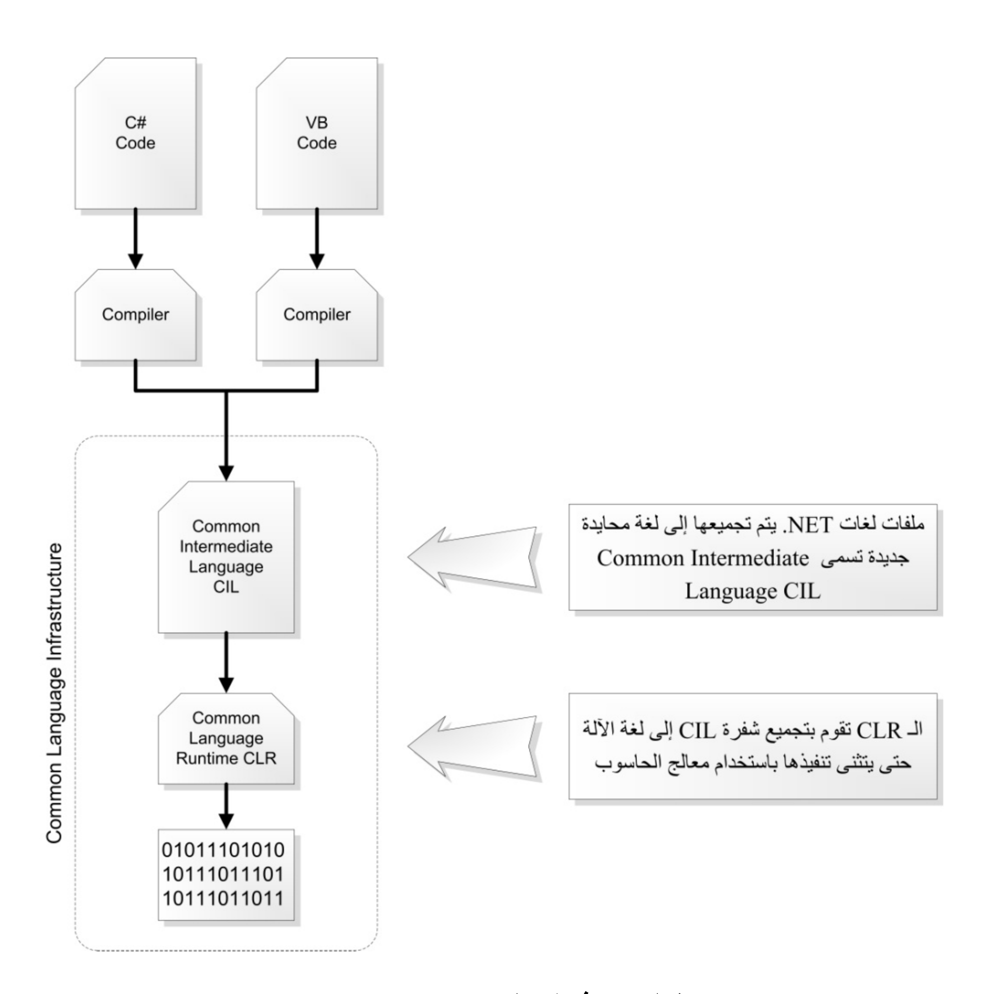
1.3 تقنية .NET Framework وكيف تعمل

في يناير ٢٠٠٢ طرحت Microsoft لأول مرة تقنية NET Framework. باعتبارها الحل الأمثل لمشاكل تطوير التطبيقات في بيئة Microsoft Windows باستخدام حزمة تطوير التطبيقات الجديدة – يومئذ – Visual Basic.NET التي طرحت للتجربة في عام ٢٠٠١.

شكل 1-1 يبين فكرة عمل تقنية .NET Framework حيث يقوم المبرمج بكتابة برنامجه بأي من لغات الحزمة Visual Studio.NET، وعندما يقوم بتجميعها، فإنه وعلى عكس اللغات التقليدية، لا يقوم المجمع بإنتاج ملف تشغيلي له الامتداد *.exe، لكنه يقوم بإنتاج نوع من الملفات يطلق عليه أسم ملفات Common Intermediate Language CIL أي ملفات اللغة الوسيطة العامة، وهي ملفات محايدة لا تتأثر بلغة البرمجة التي كتب بها التطبيق – ومن هنا وصفت بأنها محايدة – ولا يمكن أن يتم تنفيذها مباشرة – بعكس الملفات التشغيلية Executable Files التي يتم تشغيلها مباشرة – بل تحتاج لتنفيذها برنامج آخر يمثل أحد مكونات .NET Framework الأساسية وهو Common Language Runtime CLR أو منفذ اللغة العامة، وهو برنامج يلعب دور حاسوب افتراضي يمكن تنفيذ البرنامج المحفوظ في ملفات CIL بواسطته، إلا أنه في الواقع يقوم بإنشاء نسخة من البرنامج بلغة الآلة تناسب الحاسوب الحقيقي ومن ثم يقوم بتشغيلها على الحاسوب الحقيقي[1].

[1] إذا كنت تشتم رائحة Java، فأنت على حق، لقد حاولت Microsoft مضاهاة تقنية عمل تطبيقات Java.

ووفقاً لهذا السيناريو فإن تطوير التطبيق يجرى بصورة مستقلة عن نظام التشغيل والحاسوب حيث يعتمد كلياً على الـ Common Language Runtime والتي تلعب دور الوسيط بين الملفات المجمعة CIL والحاسوب. وقد وعدت Microsoft أنه قبل انتهاء عام ٢٠٠٣ سوف تكون هناك نسخة خاصة من .NET Framework لكل نظام تشغيل متعارف عليه بما فيه Linux و Unix، وبذلك يمكن استخدام لغة Visual Basic أو غيرها من لغات Visual Studio لتطوير تطبيقات تعمل على جميع أنظمة التشغيل.[1]



شكل 1-1: فكرة عمل NET Framework.

1.4تشغيل Visual Basic.NET 2008

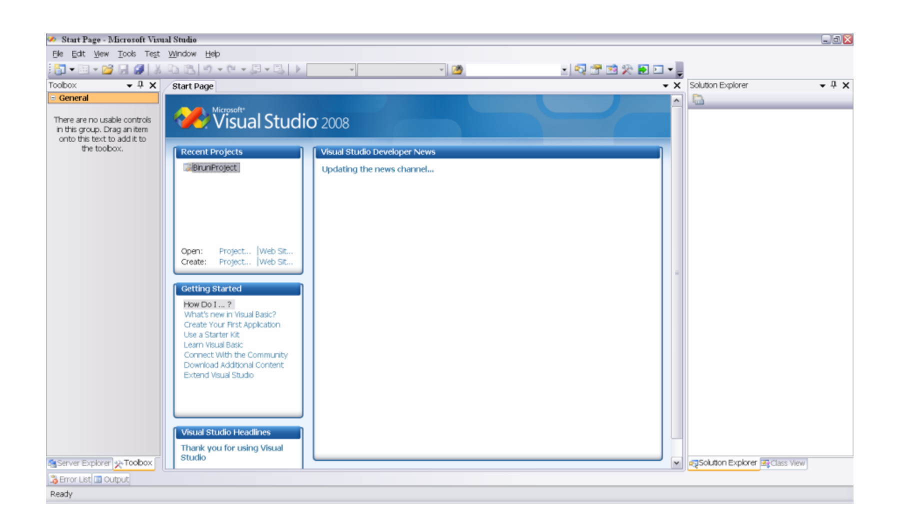
لتشغيل Visual Basic.NET 2008 يمكن أن نستخدم أحد طريقين:

من القائمة Start أختار All Programs و من هذه القائمة أختار الحزمة Visual Basic.NET 2008 ثم أختار منها Visual Basic.NET 2008.

[1] حتى اليوم فإن نسخة .NET Framework المتاحة تعمل تحت نظام التشغيل Microsoft Windows ولا توجد نسخ منها لأنظمة التشغيل الأخرى.

بالنقر فوق أيقونة Visual Basic.NET 2008 الموجودة فوق سطح المكتب.

في كلا الحالتين ينفتح Visual Basic.NET 2008 وتظهر واجهة استخدامه المبينة في شكل 2-1.



شكل 1-2: واجهة استخدام Microsoft Visual Studio عند فتحها لأول مرة

في حالة ما إذا كانت هذه هي المرة الأولى التي تقوم فيها بتشغيل Microsoft Visual Studio سوف تظهر نافذة تسأل المستخدم عن شكل واجهة التطوير المطلوب العمل معها، في هذه الحالة سوف يختار المستخدم بالطبع واجهة Visual Basic.

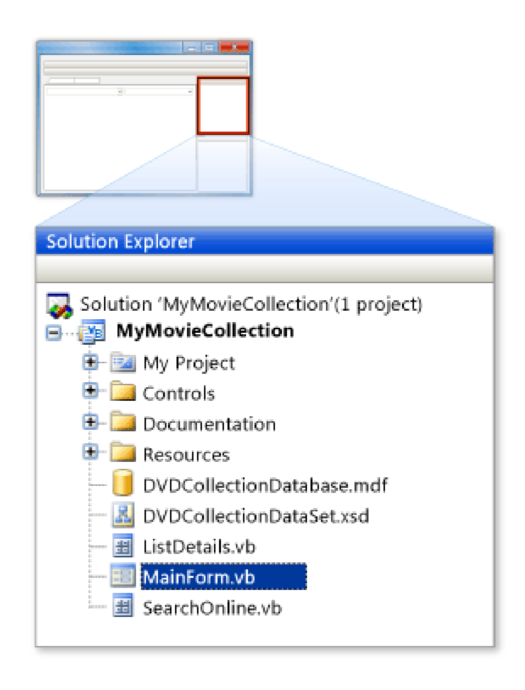
1.5 بيئة التطوير المتكاملة Integrated Development Environment IDE

بيئة التطوير المتكاملة Integrated Development Environment IDE هو الاسم الذي يستخدمه مطوري Visual Studio لوصف بيئة عملهم. وفي البداية أحب أن أقول لك أنه بالرغم من أن Visual Studio.NET 2008 يستخدم بيئة تطوير متكاملة موحدة لجميع لغات البرمجة التي يضمها، إلا أنه يفضل أن يتم تخصيص مظهر بيئة التطوير المتكاملة بما يتناسب مع Visual Basic وهو ما يمكن أن نقوم به عند تشغيل Visual Studio.NET 2008 للمرة الأولى، حيث تظهر رسالة تسأل عن الشكل الذي يرغب أن تكون عليه بيئة التطوير المتكاملة، فأحرص أن تكون Visual Basic.

تتكون بيئة التطوير المتكاملة من عدة عناصر هي:

صفحة البدء Start Page: هي صفحة وب يتم فتحها في متصفح مبسط داخل تبو يب يحمل الاسم Start Page، وتحتوي هذه الصفحة على أحدث الأخبار عن Microsoft Visual Studio.NET وكافة مكوناته، وفي حال عدم اتصال الحاسوب المثبت عليه Visual Studio بالانترنت تكون هذه الصفحة غير نشطة.

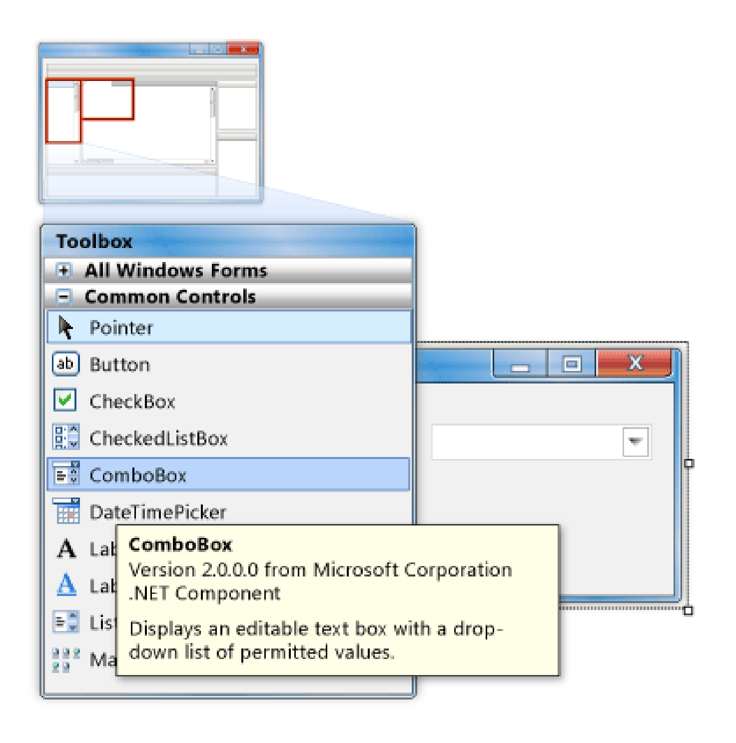
نافذة Solution Explorer: وتقع في الجانب الأيمن من بيئة التطوير المتكاملة – أنظر شكل 1-3 – حيث يمكن للمبرمج أن يطلع على مكونات الحل الذي يقوم على تطويره من مشروعات مختلفة و ما تحتويه هذه المشروعات من مكونات، كما يمكنه أن يدير جميع هذه العناصر من خلال هذه النافذة.



شكل 1-3: نافذة Solution Explorer

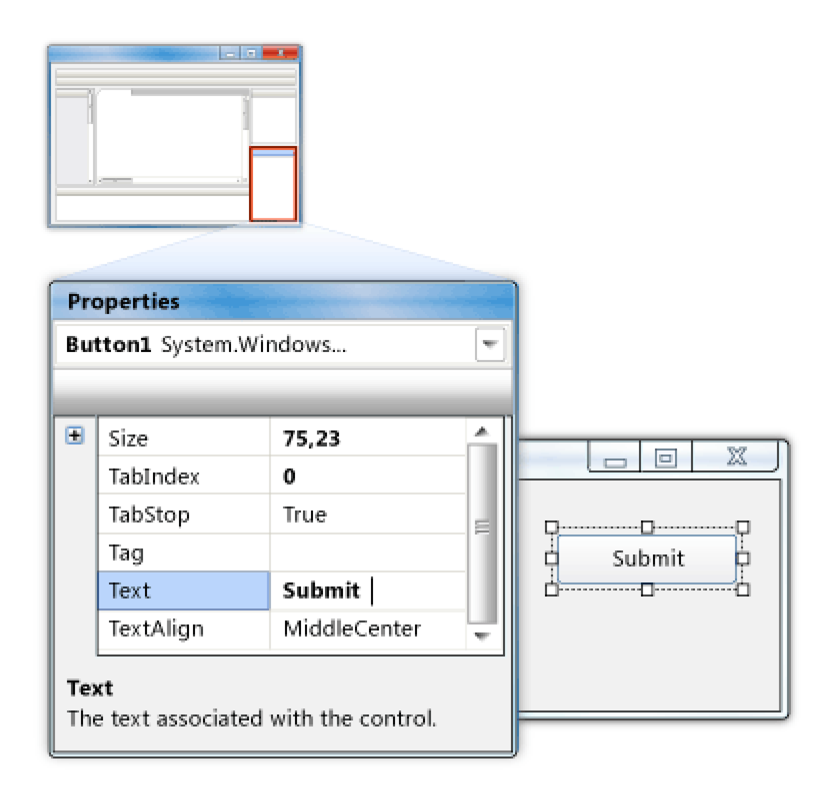
صندوق الأدوات Toolbox: و هو شريط يوجد في الجانب الأيسر من بيئة التطوير المتكاملة – أنظر شكل 1-4 – ويضم جميع عناصر التحكم Controls التي تلزم المبرمج لبناء واجهات تطبيقه مصنفه في تبويبات حيث يشمل كل تبويب مجموعة من أدوات التحكم التي تقوم بوظائف متقاربة مثلاً التبويب Menus & Toolbars يحتوي على عناصر التحكم اللازمة لإنشاء القوائم وأشرطة الأدوات و ما شابه. هناك تبويبان لا يلتزمان بجمع عناصر التحكم المتقاربة الوظيفة وهما التبويب All

Windows Forms ويضم جميع عناصر التحكم اللازمة لبناء واجهات التطبيقات، والتبويب Common Controls والي يضم عناصر التحكم الأكثر استخداماً. ويمكن من خلال اختيار أي من عناصر التحكم معرفة وظيفة هذه العنصر من خلال نافذة ملاحظة الأداة Tool Tip التي سوف تظهر.



شكل 1-4: صندوق الأدوات

نافذة الخصائص Properties Window: وهي موجودة في الركن الأيمن تحت نافذة استكشاف الحل Solution Explorer – أنظر شكل 1-5 – وهي الأداة المستخدمة لتغيير خصائص عناصر التحكم التي تتكون منها واجهة التطبيقات بصورة سهلة، حيث يتم اختيار عنصر التحكم المطلوب ضبط خصائصه، ثم اختيار الخاصية من هذه النافذة واختيار أو تخصيص قيمة ملائمة لها.



شكل 1-5: نافذة الخصائص

المشروع الأول في Visual Basic.NET

إن الطريقة المثلى لتعلم أي لغة برمجة هو الشروع فوراً في كتابة برنامج بهذه اللغة. في التدريب التالي سنقوم بكتابة برنامج بسيط الغرض منه تصفح صفحات الوب. إذا حدث ولم تفهم أي من الإجراءات المذكورة في هذا التمرين فلا تقلق فالمفاهيم المقدمة هنا سوف يتم تناولها في صورة أكثر تفصيلاً في جزء آخر من هذا الكتاب.

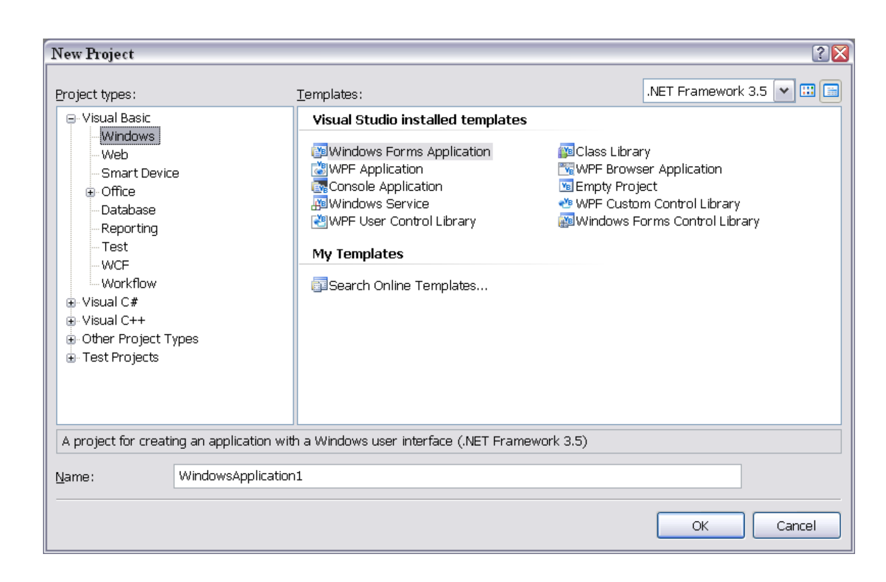
1.6إنشاء المشروع في Visual Basic.NET

الخطوة الأولى في بناء أي مشروع Visual Basic هو فتح Visual Studio وإنشاء مشروع. وهذه الخطوة ستتكرر في كل مرة تقوم فيها بالشروع في إنشاء تطبيق جديد.

لإنشاء مشروع جديد أتبع الإجراءات التالية:

من قائمة Start أختار Microsoft Visual Studio 2008 ومنها Microsoft Visual Studio 2008 لتظهر شاشة الترحيب ثم تظهر بيئة التطوير المتكاملة.

من قائمة File أختار New Project. تنفتح نافذة معنونة بـ New Project – أنظر شكل 6-2.



شكل 2-6: نافذة New Project

لاحظ أن هناك قائمة موجودة في الجانب الأيمن من هذه النافذة تحتوي على NET. Framework، يمكنك من خلال هذه القائمة اختيار الإصدارة المناسبة لـ NET. Framework وبصورة عامة إذا كنت تقوم بتطوير تطبيق ليعمل تحت نظام التشغيل Microsoft Windows Vista فلا مفر من اختيار NET Framework 3.5 . أما إذا كنت تقوم بتطوير تطبيق لن يعمل تحت هذا النظام فيمكنك اختيار أي من الإصدارات الثلاثة.

أختار Windows Application ثم أنقر OK. يظهر نموذج Form جديد في بيئة التطوير المتكاملة كما تظهر الملفات الضرورية لمشروعك في نافذة Solution Explorer. إذا كان هذا المشروع هو أول مشروع من عائلة مشروعات Windows Application فسوف يحمل الاسم WindowsApplication1.

الإجراءات السابقة كان الغرض منها إنشاء مشروع للبرنامج الذي نريد تطويره. المشروع في Visual Basic هو المكان الذي يتم اختزان وتنظيم مكونات البرنامج فيه. عند إنشاء المشروع فإنه يتم إنشاءه في ذاكرة الحاسوب، وعندما تريد إنهاء بيئة التطوير المتكاملة، فإن بيئة التطوير المتكاملة سوف تعلمك بأن المشروع غير محفوظ على الحاسوب ومن ثم فإن عليك أن تختار ما بين حفظه Save أو عدم حفظه Discard.

عند فتح النافذة المعنونة New Project – أنظر شكل 2-6 - فإنك ستجد مجموعة من المشروعات المختلفة يمكنك الانتقال من بينها. المشروع الذي قمنا باختياره والمسمى

Windows Application هو واحد من عائلة تطبيقات Windows المعتادة التي يمكنك استدعاءها من قائمة Start مثل Microsoft Word أو Internet Explorer.

عند إنشاء مشروع، يتم عرض نافذة تحتوي نموذج فارغ في بيئة التطوير المتكاملة يطلق عليها نافذة مصمم النموذج Form Designer. هذا النموذج الفارغ في مصمم النموذج يمثل النافذة التي سوف يتم عرضها في البرنامج عند تنفيذه. هناك كثير من البرامج يمكنها عرض أكثر من نافذة في نفس الوقت، لذلك فإن المشروع يمكنه أن يحتوي على أكثر من نموذج.

1.7 إنشاء واجهة الاستخدام

حان وقت إنشاء واجهة برنامج متصفح الوب. سوف تقوم ببناء واجهة استخدام برنامج باستخدام Microsoft Visual Studio 2008 وذلك بإضافة عناصر التحكم من صندوق الأدوات Toolbox إلى النموذج.

صندوق الأدوات Toolbox يقع في الجانب الأيسر من واجهة Visual Studio ويظهر في صورة لوحة Panel تظهر بالنقر فوقها وتختفي بالنقر فوق أي شئ غيرها في الواجهة، وتتكون هذه اللوحة من عدد من التبويبات Tabs مثل Data و Components و All Windows Forms. بداخل كل تبويب مجموعة من أدوات التحكم التي يمكن إضافتها إلى تطبيقك. فمثلاً، التبويب المسمى All Windows Forms يحتوي على عناصر التحكم TextBox و Button و CheckBox التي تمثل عناصر تحكم يمكن إضافتها إلى التطبيق بسحبها وإلقاءها Drag and Drop على النموذج.

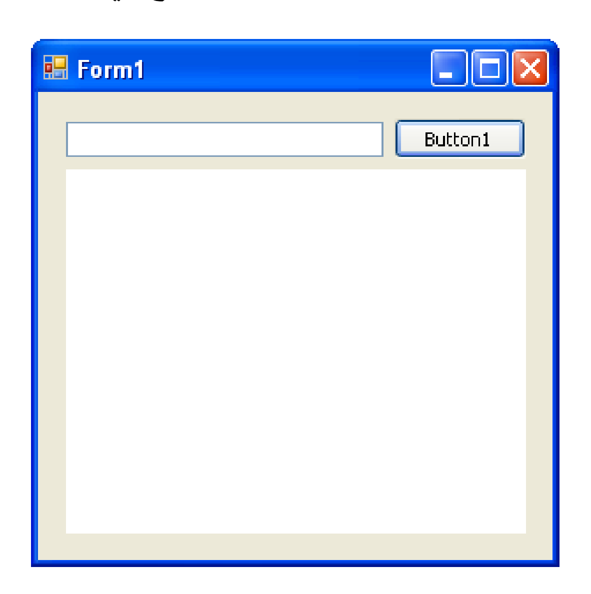
لإضافة عناصر التحكم إلى النموذج أتبع الإجراءات التالية:

1. أنقر فوق لوحة Toolbox لتظهر هذه اللوحة.
2. أنقر تبويب All Windows Forms ثم أختار عنصر التحكم Panel ثم اسحب وألقي هذا العنصر في الركن الأعلى اليسر من النموذج.
3. من نفس التبويب، أسحب عنصر التحكم Button ثم ألقيه فوق عنصر التحكم Panel الذي سبق أن قمت بإلقائه فوق النموذج.

4.من نفس التبويب اسحب عنصر التحكم TextBox ثم قم بإلقائه فوق Panel الموجود فوق الــ Form في جانبه الأعلى.

5.وأخيراً من نفس التبويب أخذ تر WebBrowser وضعه في الجانب الأسفل للــ Panel.


من المفترض أن تحصل على شكل مشابه للموضح في



شكل 2-7: النموذج بعد إضافة عناصر التحكم

الآن أنت قد أضفت عناصر التحكم إلى النموذج. عناصر التحكم تحتوي على شفرة Code يعرف مظهر ومهام العنصر. فمثلاً، العنصر Button، عادة معظم واجهات التطبيقات تحتوي مفتاح OK أو Exit، وفي الماضي عند تصميم مثل هذا المفتاح كان الأمر يستلزم كتابة شفرة حول كيف سيبدو المفتاح وكيف سيتغير مظهرة بالنقر عليه وهكذا، وهذه مسألة صعبة ومضيعة للوقت، لكن باستخدام Visual Basic فإن العنصر Button يحتوي على الشفرة الأساسية التي تضبط هيئته ليوفر بذلك جانب ثمين من الوقت.

كما ترى فإن Toolbox يحتوي عدد كبير من عناصر التحكم، ولكل عنصر من هذه العناصر وظيفة واحدة. العنصر Panel مثلاً يمكن أن يستخدم كحاوية لغيره من العناصر. بينما يستخدم العنصر Button للقيام بتنفيذ إجراءات محددة عندما يقوم المستخدم النهائي للتطبيق بالنقر عليه. أما عنصر التحكم TextBox فيستخدم لإدخال البيانات النصية من لوحة المفاتيح وإظهارها فوق شاشة الحاسوب. وأخيراً عنصر التحكم WebBrowser يقدم خصائص تصفح الوب مماثلة لتلك التي يقدمها Internet Explorer.

1.8ضبط خصائص عناصر التحكم

قمنا فيما سبق بإنشاء واجهة التطبيق عن طريق إضافة عناصر التحكم إلى النموذج. حتى هذه اللحظة فإن النموذج لا يظهر بصورة تناسب ما هو معرف من التطبيقات كما لا تقوم عناصر واجهته بأي من الوظائف. فيما يلي نعتني بالجزء الأول من المشكل وهو مظهر التطبيق حيث سوف نقوم بضبط الطريقة التي يبدو عليها البرنامج ليظهر جيداً وذلك بضبط خصائص عناصر التحكم.

لضبط خصائص عناصر التحكم نتبع الإجراءات التالية:

6.أختر عنصر التحكم Panel من فوق النموذج في نافذة Form Designer بالنقر فوقه أو بالنقر فوق أي من حوافه. تعرض نافذة Properties في القسم الأسفل الأيمن من بيئة التطوير المتكاملة خصائص عنصر التحكم Panel.

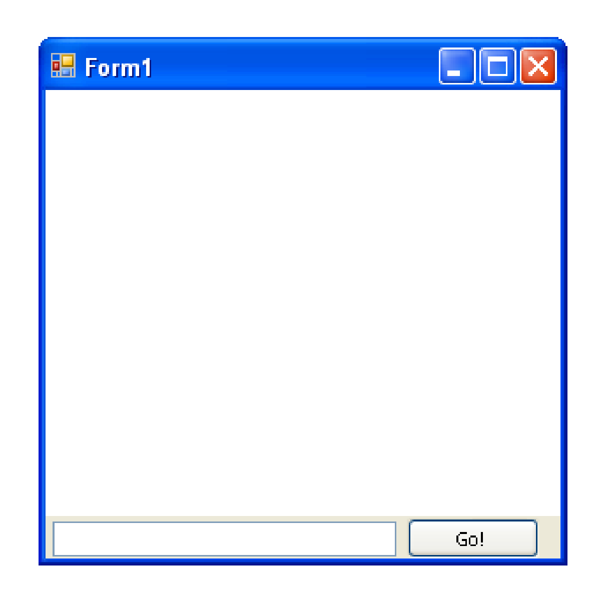
7. في نافذة Properties أختر خاصية Dock ثم انقر السهم الموجود على يمين الخاصية، تظهر نافذة صغيرة تحتوي على عدد من الصناديق.

8.انقر فوق الصندوق الأعلى لتضبط خاصية Dock على أعلى، ونتيجة لهذا يتمدد عنصر التحكم Panel ليشغل القسم العلوي من النموذج.

9.قم باختيار عنصر التحكم WebBrowser من فوق النموذج. في نافذة Properties أختر خاصية Dock ثم اضبطها لتصبح Fill (الصندوق الأوسط).

10.أختر عنصر التحكم Button ومن نافذة Properties أختر خاصية Text ثم امح كلمة Button1 من أمامها وأكتب عوضاً عنها Go!.

11.قم بتغيير وضع العناصر وتغيير حجمها بالنقر عليها وجرها من مكانها أو النقر على حوافها وسحب هذه الحواف حتى تحصل على الشكل التالي للواجهة.



شكل 2-8: شكل النموذج بعد ضبط عناصر التحكم

يمكن التحكم في شكل عناصر التحكم عن طريق تغيير ضبط خصائصها. فمثلا تغيير قيمة الخاصية Text لعنصر التحكم Button أدى إلى تغير العنوان المكتوب فوق المفتاح. وتتغير قيم خصائص عناصر التحكم بصدور شتى، فمنها ما يتغير رقمياً ومنها يحصل على قسمة نصية من قائمة محددة، ومنها ما له قيمتين False و True.

يمكن تغيير الخصائص من النافذة Properties وعندئذ يتغير شكل عنصر التحكم لملائمة القيمة الجديدة لخصائصه، كما يمكن تغيير الخصائص أثناء التصميم كتغيير الموقع أو تغير الحجم و في هذه الحالة تتغير قيمة الخصائص المقابلة في النافذة Properties لتتوافق مع هذه التغيرات.

نظراً لأن عملية ضبط الخصائص ستتكرر كثيراً فيما يلي، فسوف نستخدم طريقة مبسطة لشرح خصائص عناصر التحكم المختلفة حيث سوف نستخدم جدول مكون من ثلاثة أعمدة، العمود الأول نذكر فيه اسم عنصر التحكم، والعمود الثاني سنذكر فيه الخاصية وفي العمود الثالث وفي مقابل الخاصية نضع قيمتها.

والجدول التالي يبين ضبط خصائص النموذج Form1 الذي قمنا به من قبل:

عنصر التحكم	الخاصية	القيمة
Panel	Dock	Top
WebBrowser	Dock	Fill
Button	Test	Go!

1.9إضافة شفرة البرنامج

فيما سبق قمنا بتعديل مظهر عناصر التحكم لنحصل على مظهر ملائم للتطبيقات، فيما يلي نعني بما يجعل هذا التطبيق له القدرة على العمل والاستجابة لطلبات مستخدميه.

قم بإضافة شفرة للبرنامج بالطريقة التالية:

12.قم بالنقر مرتين فوق المفتاح Button المسمى Go! فوق النموذج. تنفتح نافذة جديدة معنونة Code Editor في تبويب جديد.

13.قم بكتابة الشفرة التالية في نافذة Code Editor.

```
WebBrowser1.Navigate(TextBox1.Text)
```

14.هذه الشفرة سوف تعمل فقط عند تنفيذ البرنامج.

هل لاحظت أن نافذة Code Editorعندما ظهرت كانت تحتوي على الشفرة التالية:

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByValue As   System.EventArgs) Handles
Button1.Click
|
End Sub
```

هذه الشفرة يطلق عليها إجراء جزئي Sub Procedure. أي شفرة بداخل هذا الإجراء الجزئي (أي بين العبارات Sub و End Sub) سوف يتم تنفيذها في كل مرة يتم النقر على المفتاح فيها.

الشفرة التي قمت بكتابتها

(WebBrowser1.Navigate(TextBox1.Text

تفيد البرنامج أن يستخدم الطريقة method المسماة Navigate الخاصة بعنصر التحكم WebBrowser (و هو المسمى هنا WebBrowser1) مستخدم القيمة النصية المرتجعة من عنصر التحكم TextBox1 المشار إليها بـ- TextBox1.Text حتى يقوم بتحميل الصفحة التي كتب المستخدم عنوانها في عنصر التحكم TextBox في عنصر التحكم عند النقر على المفتاح Go!.

1.10اختبار وتشغيل البرنامج

الآن وقد أنهينا البرنامج قد حان الوقت لاختباره وتنفيذه. في حالة التطبيقات لضخمة والمعقدة قد يكون اختبار التطبيق طويلاً ومعقداً، و هو الأمر الذي سوف نناقشه في موضع آخر. أما في حالة البرنامج البسيط الذي قمنا بتطويره كل ما نحتاجه أن نقوم بتشغيله فقط.

لتشغيل البرنامج قم بالإجراءات التالية:

15. قم بتوصيل الحاسوب إلى الانترنت.

16. من القائمة Debug أختر Start Debugging ليبدأ تشغيل البرنامج.

17. قم بكتابة عنوان أي من مواقع الوب في صندوق النصوص في البرنامج وليكن www.google.com ثم أنقر المفتاح Go!. تظهر صفحة الموقع في عنصر التحكم WebBrowser.

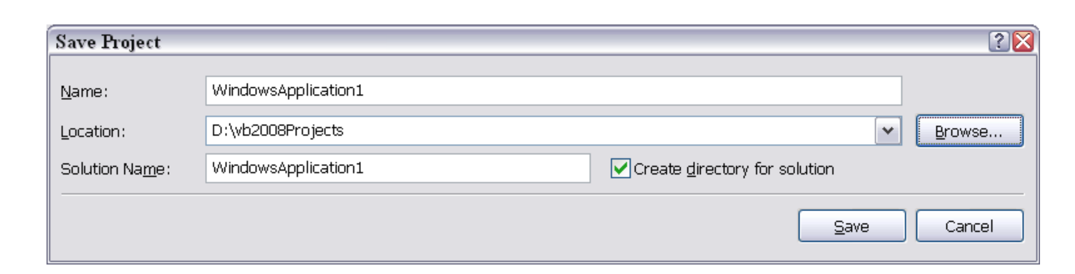
18. لإنهاء تشغيل البرنامج أختار Stop Debugging من القائمة Debug.

في معظم التطبيقات التي تقوم بتطويرها سوف تقوم باختبار التطبيق أثناء تطويره العديد من المرات، ففي كل مرة سوف تضيف فيها عنصر تحكم أو تضيف شفرة سوف ترغب أن تعلم ما إذا كان التطبيق يعمل على الوجه المطلوب أم لا. هذه العملية يطلق عليها أسم الـ Debugging وسوف نناقشها لاحقاً.

1.11حفظ المشروع

الآن أنت قد أنشأت تطبيقك الأول. لنقوم بحفظه أتبع الخطوات التالية:

19. من القائمة File أختار Save All لتظهر النافذة المبينة في شكل 2-9.



شكل 2-9: نافذة Save Project

في الخانة Name يمكنك تعديل اسم البرنامج.

من الخانة Location أكتب عنوان المجلد الذي تريد أن يضم مشروعك.

في الخانة Solution Name قم بكتابة أو تعديل اسم الحل.

في النهاية أنقر فوق المفتاح Save.

يجب أن نتوقف قليلاً لنعلق على موضوعين.

الموضوع الأول في هذا النافذة يو جد خا نة للا سم Name وخا نة لأ سم الـ حل Solution Name، فما الفارق بينهما؟ الاسم هو الاسم الذي سوف يظهر به التطبيق عند أضافته لمجو عة البرامج خاصتك التي يدير ها Microsoft Windows أو بصورة أكثر بساطة هو اسم تطبيقك كما سوف يظهر في القائمة All Programs. بينما اسم الحل فهو ا سم الملف الذي سوف يسـ تخدم في إنشاء ملف الت طبيق أثناء عمل ية تطو ير الت طبيق في Visual Basic.NET وهو ملف سوف يكون له الامتداد *.sln.

الموضوع الثا ني يتعلق بالمكان الذي سوف يـتم حـفظ المشروع علـيه أثناء تطويره. فالمشروع قبل عملية الحفظ أين كانت ملفاته؟ في الواقع كانت هذه الملفات موجودة في مجلد خاص بعنوان Visual Studio 2008. لكن عند حفظ المشروع فإنه يمكنك تخصيص مجلد مخصوص لحفظ هذا المشروع وهو المجلد الذي سوف تحدد مساره في الخانة Location.

أساسيات Visual Basic.NET 2008

يعتبر Microsoft Visual Basic 2008 بمختلف إصداراته الطريقة الأسرع والأسهل لتطوير التطبيقات في بيئة تشغيل Microsoft Windows. وحتى هؤلاء المبرمجين المبتدئين، فإن Visual Basic يمدهم بمجموعة كاملة من الأدوات التي تيسر أي من عمليات تطوير التطبيقات.

ولكن ما هو Visual Basic؟

إن اللفظ Visual يشير إلى الطريقة التي يقوم بها المبرمج بتطوير واجهات الاستخدام الرسومية (Graphical User Interface (GUI لتطبيقاته. أما اللفظ Basic فهو يشير إلى الأحرف الأولى من العبارة Beginners App-Purpose Symbolic Instruction Code بمعنى شفرة التعليمات الرمزية لمختلف أغراض المبتدئين، لغة البرمجة التي أستخدمها أكبر عدد من مبرمجي الحاسوب منذ بداية الحاسوب. باستخدام Visual Basic يمكنك تطوير العديد من التطبيقات النافعة بأقل قدر من التعقيد.

في هذا القسم نتتبع معاً أهم عناصر لغة برمجة Visual Basic 2008.

1.12 ما هي البرمجة؟

ربما كان من المستحسن قبل أن نمضي في تعلم لغة برمجة Visual Basic أن نتوقف قليلاً لنفهم ما هي لغة البرمجة Programming Language وكيف تعمل. أي أن علينا أن نعلم كيفية البرمجة من الجانب النظري.

الحاسوب في حد ذاته ليس ذكياً بالمرة.

الحاسوب في حد ذاته مجرد حزمة ضخمة من المكونات الإلكترونية التي تمرر التيار الكهربي أو توقفه، لكن المستخدم هو من يمكنه أن يضبط هذه المكونات الإلكترونية بحيث يجعل الحاسوب يعرض صورة أو يعزف قطعة من الموسيقى، وهذا هو البرمجة في أبسط صورها، أن تجعل الحاسوب ينفذ أمر ما.

بالطبع إن عملية ضبط هذه المكونات الإلكترونية هي عملية مجهدة ومعقدة، وهنا يأتي دور لغات البرمجة.

1.12.1 ما هي لغة البرمجة؟

يعبر الناس عن أنفسهم أستخدم لغات تتكون من العديد من الكلمات. الحاسوب يستخدم لغة بسيطة تتكون من حرفين فقط ١ و ٠. فيعبر عن وجود التيار الكهربي في دائرته بالعدد ١ و عن انعدام وجود التيار الكهربي بالعدد ٠. إن محاولة التخاطب مع الحاسوب باستخدام لغته أشبه بالتخاطب مع صديق باستخدام شفرة مورس. ممكن؟ نعم لكن صعب.

تعمل لغات البرمجة بمثابة مترجم بينك وبين الحاسوب. فعوضاً عن تعلم اللغة الأصلية للحاسوب – والتي يطلق عليها أسم لغة الآلة Machine Language – يمكن استخدام لغات البرمجة بطرقة أبسط وأكثر قابلية للفهم.

عند كتابة برنامج بأي من لغات البرمجة، يقوم برنامج خاص يسمى المجمع compiler بتحويل التعليمات التي كتبت بلغة البرمجة إلى لغة الآلة. وهذا يعني أن مبرمج Visual Basic ليس عليه أن يهتم بكيف سيقوم الحاسوب بتنفيذ تعليماته، بل عليه أن يعرف كيف يعبر عما يرغب باستخدام Visual Basic.

1.12.2 طبيعة لغة Visual Basic.NET

تتشابه لغة Visual Basic مع اللغة التي نستخدمها في حياتنا اليومية. عندما نتحدث أو نكتب نستخدم أنواع مختلفة من الكلمات مثل الأفعال والأسماء، والتي تستخدم بطريقة معرفة سلفا. كذلك في Visual Basic هناك مجموعة من الكلمات المعروفة باسم عناصر البرمجة المعرفة الاستخدام والتي تستخدم في كتابة البرامج.

تشتمل عناصر البرمجة في Visual Basic على العبارات Statements والإعلانات Declarations والطرق Methods والعاملات Operators والكلمات المحجوزة Keywords. بنهاية هذا الكتاب سوف تكون ملم بهذه العناصر وكيفية استخدامها.

اللغات الإنسانية لها قواعد Syntax التي تحدد كيف يتركب الكلام في جملة. وكذلك Visual Basic تحتوي على قواعد خاصة بها تتميز بالسهولة الشديدة. فمثلاً العبارة "السرعة القصوى لسيارتي هي ٥٠" يمكن التعبير عنها في Visual Basic كما يلي:

```
Car.Speed.Maximum=50
```

في نهاية هذا القسم ستتعلم الكثير عن قواعد لغة Visual Basic والأدوات التي سوف تساعدك في تصحيح صيغ الشفرة في برنامجك مثل الأداة IntelliSense التي تساعد المبرمج في تصحيح شفرة البرنامج أثناء كتابته إياها.

وكما أن اللغة البشرية تتكون من أجزاء، فمثلاً يتكون هذا الكتاب من أبواب وبكل باب عدد من المقاطع يتكون كل مقطع من جمل، أيضاً تتكون البرامج المكتوبة بـ Visual Basic من أجزاء يطلق عليها modules التي تقوم مقام الباب، و procedures التي تقوم مقام المقاطع، وأخيراً سطور الشفرة lines of code التي تقوم مقام العبارات.

1.12.3الخصائص والطرق والأحداث

عندما قمنا بإنشاء برنامج متصفح الانترنت في القسم رقم ٢ من هذا الكتاب، استخدمنا عناصر التحكم مثل المفتاح Button وصندوق النصوص TextBox لبناء واجهة البرنامج. تمثل عناصر التحكم التي استخدمناها نموذج مثالي لما يطلق عليه في Visual Basic.NET اسم الكائن Object.

وبالرغم من أننا سنتعرض للكائنات في قسم مستقل من هذا الكتاب، إلا أن عرض مبسط للمفاهيم الأساسية للكائنات سيبسط لنا الكثير من المسائل التي سوف نتناولها في هذا القسم وأقسام أخرى قادمة.

الكائنات Objects كما تقدمها Visual Basic لمستخدميها هي محاكاة لمفهوم الكائنات في الواقع الحي. فأي كائن في الواقع الحي يمكن التعامل معه من خلال ثلاثة مفاهيم هي الخصائص Properties والطرق Methods والأحداث Events.

خصائص الكائن هي مجموعة الأوصاف التي تميزه، فالبالون ككائن في الواقع يمكن وصفه عن طريق بعض الخصائص مثل اللون والقطر وحالته من حيث كونه منتفخاً أم فارغ. فإذا انتقلنا إلى Visual Basic فإن هذه الخصائص يمكن ترجمتها إلى شفرة تعليمات كالتالي:

```
Balloon.Color = Red
Balloon.Diameter = 10
Balloon.Inflated = True
```

يمكن أن نقرأ هذه السطور من الشفرة كالتالي: خاصية اللون للكائن بالون هي أحمر، خاصية القطر Diameter للكائن بالون تساوي ١٠، خاصية الانتفاخ Inflated للكائن بالون حقيقية (أي أن البالون منتفخ).

بصورة عامة فإن خاصية أي كائن في Visual Basic يمكن تمثيلها كالتالي:

```
object.property = value
```

حيث object اسم الكائن، property الخاصية التي نقوم بضبطها، value هي القيمة التي سوف نخصصها للخاصية. ويسمى ضبط الخاصية بهذه الطريقة باسم الضبط بالشفرة البرمجية تمييزاً له عن الضبط عن طريقة تغيير قيمة الخاصية في النافذة Properties كما فعلنا في القسم الثاني من الكتاب. من الناحية العلمية كلا الطريقتين يؤديان إلى نفس النتيجة، إلا أن طريقة الضبط بالشفرة البرمجية هي الطريقة الوحيدة التي يمكن استخدامها عند تصميم الاستجابة عن حدث معين كما سوف يأتي.

المفهوم الثاني الذي يتعلق بالكائن هو سلوك الكائن، فأي كائن في العالم يمكن أن يتصرف تصرفات عدة، فالكلب مثلاً ككائن يمكن أن ينبح أو يحرك ذيله، وكذلك البالون يمكنه أن يقوم بعدد من التصرفات كأن ينتفخ أو يفرغ الغاز الذي يحتويه أو يرتفع إلى أعلى. تسمى هذه التصرفات أو السلوك في Visual Basic بالاسم "طرق Methods" كما بعض اللغات الأخرى تدعوها "وظائف Functions" أي الوظائف التي يمكن أن يقوم بها الكائن.

الآن لو أننا حاولنا أن نمثل الوظائف التي يقوم بها البالون مستخدمين Visual Basic.NET كيف سنعبر عن ذلك؟ سنعبر عنها في صورة شفرة تعليمات كما يلي:

```
Balloon.Inflate
Balloon.Deflate
Balloon.Rise(5)
```

وهو ما يمكن قراءته كما يلي. البالون يقوم بوظيفة الانتفاخ Inflate، البالون يقوم بوظيفة أفراغ غازه Deflate، البالون يقوم بوظيفة الارتفاع إلى أعلى مسافة خمسة.

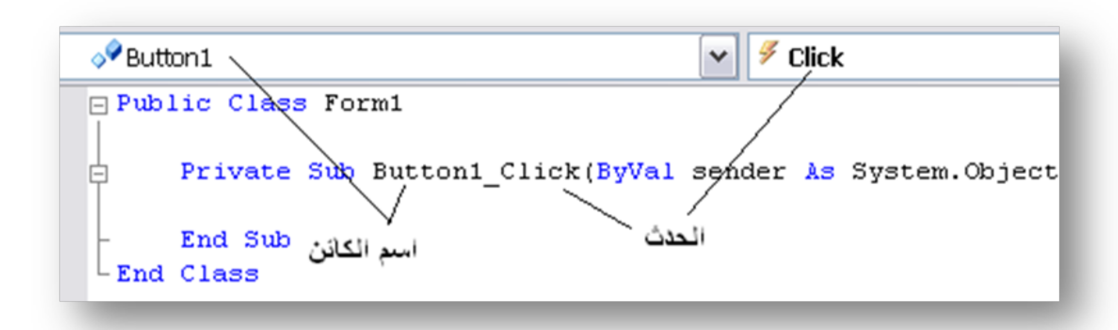
وبصورة عامة يمكن تمثيل قيام كائن بطريقة ما باستخدام الصيغة:

```
object.method (parameter1,parameter2,..)
```

حيث object اسم الكائن، method اسم الطريقة المطلوب منه تنفيذها، parameter1 و parameter2 وجميع البارامترات الموجودة بين الأقواس المستديرة () هي قيم أو كائنات أخرى تخصص عمل الطريقة.

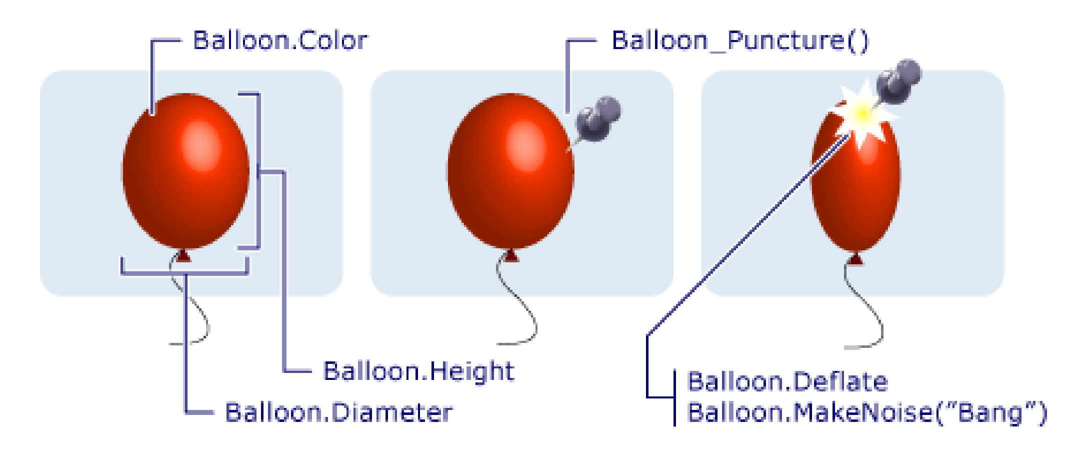
المفهوم الثالث الأساسي بالنسبة للكائنات – وهو في الواقع أهم هذه المفاهيم الثلاثة – هو مفهوم الحدث Event. فأي كائن قد يتعرض لي حدث خارجي ومن ثم يستجيب إلى هذا الحدث، فلو انك وخزت البالون بإبرة سوف تنفجر، فالوخز هو الحدث، والانفجار هو استجابة البالون لحدث الوخز.

في Visual Basic توجد مجموعة من الأحداث التي يمكن أن يستجيب لها أي كائن. و يمكن معرفه هذه الأحداث من خلال نافذة كتابة الشفرة كما هو مبين في شكل 10-3.



شكل 3-10: التعرف على الأحداث من خلال نافذة كتابة شفرة التعليمات

الآن لنفترض أن البالون الذي نحاول تمثيله في Visual Basic تعرض للوخز Puncture، كيف يمكن أن نقوم بتمثيل هذا الحدث في Visual Basic؟. شكل 3-11 يقدم لنا فكرة مصورة عن الطريقة التي سوف يتم تمثيل هذا الحدث بها.



شكل 3-11: حدث وخز البالون

شفرة التعليمات التالية تمثل استجابة البالون لحدث الوخز.

```
Sub Balloon_Puncture()
   Balloon.MakeNoise("Bang")
   Balloon.Deflate
   Balloon.Inflated = False
End Sub
```

بدا ية نلا حظ أن الس طر الأول يتكون من الكل مة Sub يتبع ها Balloon_Puncture()، هذا السطر يمثل بداية الحدث، وجميع التعليمات التي ستأتي بعد هذا السطر تمثل استجابة الكائن Balloon لحدث الوخز، بينما البارامترات الموجودة بين القوسين () – في حالتنا هذه لا توجد بارامترات – تخصص طريقة الحدث.

يتكون الحدث من مجموعة من السطور تمثل تغيير في قيم بعض خصائص الكائن Balloon وحثه على القيام ببعض الوظائف أو الطرق. فمجموع تعليمات الاستجابة على الحدث يم كن قراءتها هكذا. ع ند وقوع حدث الو خز، سيستخدم الكائن بالون الطر قة MakeNoise أي طريقة إصدار ضوضاء ويتم تخصيص الضوضاء في شكل كلمة Bang التي تقدم كبارامتر بين القوسين ()، ثم يقوم الكائن بتنفيذ طريقة Deflate فيقوم بتفريغ نفسه من الغاز الذي يملؤه، وأخيراً تصبح الخاصية Inflated بمعنى ممتلئ بالغاز غير حقيقية. وهذا هو ما يمثل استجابة البالون لحدث الوخز.

في النهاية يأتي السطر الأخير و هو End Sub ليمثل نهاية الحدث، وتعتبر جميع التعليمات الواردة بين Sub Balloon_Puncture()، و End Sub هي استجابة الكائن Balloon على حدث الوخز.

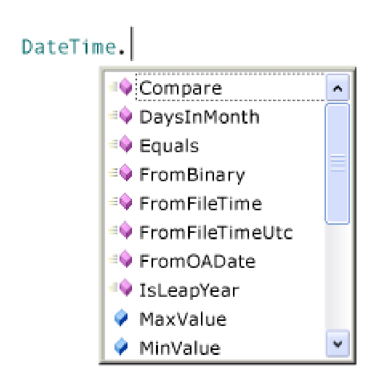
1.12.4 الوظيفة IntelliSense

أهم ميزات Visual Studio.NET 2008 IDE في كتابة البرامج هو البساطة والسهولة، حيث يمكن كتابة برامج طويلة باستخدام عدة ضربات قليلة على لوحة المفاتيح لكتابة تعليمات البرنامج وبأقل قدر ممكن الأخطاء، وذلك بفضل الوظيفة IntelliSense.

تقدم هذه الوظيفة عدد من الأدوات التي تساعد المبرمج على كتابة تعليمات شفرة برامجه ببساطة وبدون أخطاء، تشتمل هذه الأدوات على سرد الأعضاء List Members، معلومات البارامترات Parameter Info، والمعلومات السريعة Quick Info، وإكمال الكلمات Complete Word، وملاحظات البنى اللغوية Syntax Tips.

1.12.4.1 سرد الأعضاء List Members

عند ما تقوم بكتابة اسم كائن متبوعاً بمؤشر النقطة point operator، فإن قائمة بجميع خصائص وطرق وأحداث – و هي ما يطلق عليها أسم أعضاء الكائن على ما سوف نعرف في قسم متقدم من هذا الكتاب - هذا الكائن سوف تسرد كما هو مبين في شكل 3-12، حيث يمكن التحرك فيها باستخدام الفأرة أو باستخدام مفاتيح الأسهم، كما يمكن الاختيار منها عن طريق ضرب مفتاح Space في لوحة المفاتيح حيث سوف يتم إضافتها على جوار الكائن فوراً عقب مؤشر النقطة.



شكل 3-12: سرد الأعضاء

1.12.4.2معلومات البارامتر *Parameter Info*

عندما تكتب طريقة ما وتعقبها بفتح قوس مستدير) لإضافة البارامترات خاصتها، تظ هر نافذة تحتوي على معلومات حول هذه البارامترات المطلوب إضافتها، وعند ما يكون هناك أكثر من أسلوب لو ضع هذه البارامترات بين الأقواس المستديرة يمكنك استعراض الأساليب المختلفة عن طريق تحريك بكرة الفأرة أو استخدام مفاتيح الأسهم. أنظر شكل 3-13 لمزيد من الإيضاح.

```
Console.Write(
▲1 of 18  ▼ void Console.Write (bool value)
value: The value to write.
```

شكل 3-13: معلومات البارامتر

1.12.4.3المعلومة السريعة *Quick Info*

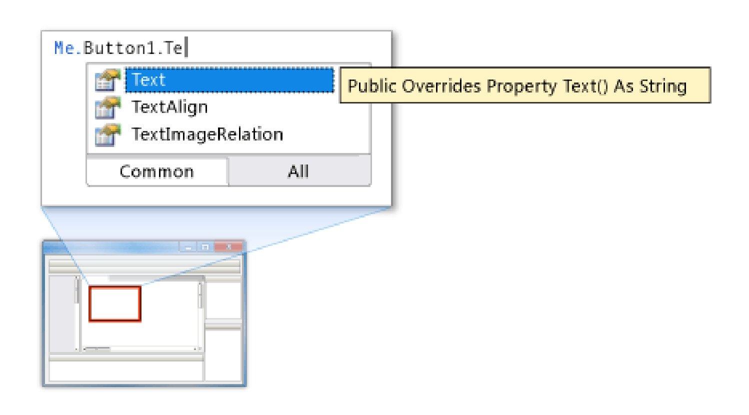
إذا قمت بإسناد مؤشر الفأرة فوق أي كلمة من الكلمات التي تكون شفرة التعليمات سوف يظهر مربع يحتوي على تعريف كامل لماهية هذه الكلمة كما هو مبين في شكل 3-14.

```
DateTime.Compare(FirstDate, SecondDate);
int DateTime.Compare (DateTime t1, DateTime t2)
Compares two instances of System.DateTime and
returns an indication of their relative values.
```

شكل 3-14: المعلومة السريعة

1.12.4.4إكمال الكلمات *Complete Word*

عندما تقوم بكتابة اسم لخاصية او طريقة أو حدث تظهر قائمة تستكمل لك هذا الاسم كما هو مبين في شكل 3-15 يمكن إدخال الاسم منها مباشرة بضرب المفتاح Space من لوحة المفاتيح.



شكل 3-15: إكمال الكلمات

1.13 تمثيل البيانات باستخدام المتغيرات

تمثل المتغيرات Variables مفهوم أساسي في برمجة الحواسيب. المتغير هو حرف أو أسم يتم اختزان القيمة فيه. عندما تقوم بإنشاء برنامج يمكنك استخدام المتغيرات لاختزان الأرقام مثل ارتفاعات الأبنية أو الكلمات مثل أسماء الأشخاص. بصورة عامة يمكن استخدام المتغيرات لتمثيل أي نوع من البيانات مطلوب في برنامجك.

قد تسأل نفسك سؤال "لماذا أستخدم المتغيرات في حين يمكن استخدام قيم البيانات مباشرة؟". كما هو واضح من اللفظ "متغيرات Variables" فإن القيم التي تختزن في المتغيرات يمكن أن تتغير أثناء تشغيل البرنامج. فعلى سبيل المثال يمكنك أن تكتب برنامج لتعقب عدد علب المياه الغازية في الثلاجة، هذا العدد يتغير مع الوقت حيث أن هناك من يشرب هذه العلب وهناك أيضاً من يضيف علب جديدة إلى الثلاجة، وعوضاً عن كتابة كل البيانات الخاصة بتغير عدد علب المياه الغازية داخل البرنامج، يمكنك أن تستعيض عنها بمتغير يتم تزويده بعدد العلب أثناء تشغيل البرنامج.

1.13.1 اختزان البيانات في المتغيرات

هناك ثلاثة خطوات لاستخدام المتغيرات، هذه الخطوات الثلاثة هي:

الإعلان عن المتغير Declare the Variable

تخصيص المتغير Assign the Variable

استخدام المتغير Use the Variable

1.13.1.1 الإعلان عن المتغيرات

عند الإعلان عن متغير يجب أن تحدد أسم لهذا المتغير ونوعية البيانات التي سوف يختزنها. تستخدم للإعلان عن المتغيرات الكلمة المحجوزة Dim متبوعة باسم المتغير ثم الكلمة المحجوزة As متبوعة بنوع البيانات التي سوف يتم اختزانها في هذا المتغير.

```
Dim aNumber as Integer
```

هذا السطر من الشفرة يخبر البرنامج أنك تريد الإعلان عن استخدام متغير أسمه aNumber ليختزن بيانات نوعها Integer. ونتيجة لتعريف المتغير aNumber كمتغير من النوع Integer فهو يمكنه فقط اختزان قيم رقمية صحيحة فقط.

1.13.1.2 تخصيص المتغير

لتخصيص قيمة ليتم اختزانها في المتغير تستخدم علامة = والتي قد تدعي أحيانا معامل التخصيص assignment operator.

```
aNumber = 42
```

هذا السطر من الشفرة يقوم باختزان الرقم ٤٢ داخل المتغير المسمى aNumber.

1.13.1.3 الإعلان عن المتغيرات وتخصيص قيم افتراضية لها

كما تعرفنا سابقاً يمكننا الإعلان عن المتغير في سطر ثم تخصيص قيمته في سطر آخر، وهذا ما قد ينتج عنه خطأ إذا أردت أن تقوم بتخصيص قيمه المتغير قبل الإعلان عنه.

لهذا السبب فمن المستحسن أن يتم الإعلان عن المتغير وتخصيص قيمته في سطر واحد. حتى ولو لم تكن تعرف ما هي القيمة التي سوف تختزن في المتغير يمكنك تخصيص قيمة افتراضية. السطر التالي يبين كيف يمكن للمستخدم أن يختزل السطرين المذكورين في 1.13.1.1 1.13.1.1 و 1.13.1.2 مرة واحدة.

```
Dim aNumber As Integer = 42
```

1.13.1.4 تدريب: الإعلان عن المتغيرات وتخصيص قيم لها

في هذا التدريب سوف نحاول أن نقوم بكتابة برنامج يوضح ما سبق حيث يتم الإعلان عن المتغيرات وتخصيص قيم لها، ثم إظهارها على الشاشة في صندوق رسائل Message Box. بداية قم بإنشاء مشروع جديد متبعاً الخطوات التالية:

20. قم بفتح Visual Basic من قائمة Start.

21. من قائمة File أختر New Project.

22. من القوالب الظاهرة في نافذة New Project أختر WindowsApplication.

23. قم بتخصيص أسم للمشروع في الخانة Name وليكن Variables.

قم بكتابة شفرة البرنامج متتبعاً الخطوات التالية:

24. أنقر مرتين فوق النموذج المعروض. تظهر نافذة تحرير الشفرة Code Editor على مقطع من الشفرة معنون بـ Form_Load. هذا المقطع وأترابه يطلق عليه أسم الإجراء Procedure وهو يحتوي التعليمات التي سوف تنفذ عند تحميل النموذج إلى ذاكرة الحاسوب.

25. في الإجراء Form_Load قم بكتابة الشفرة التالية:

```
Dim anInteger As Integer = 42
Dim aSingle As Single = 39.345677653
Dim aString As String = "I like candy"
Dim aBoolean As Boolean = True
```

يعرف هذا الكود أربعة متغيرات من أربعة أنواع مختلفة هي anInteger متغير من النوع Integer و aSingle متغير من النوع Single و aString متغير من النوع String و aBoolean متغير من النوع Boolean ويتم تخصيص قيم مناسبة لهم كلاً في سطر تعريفه.

قم بكتابة الشفرة التالية أسفل الشفرة السابقة.

```
MsgBox(anInteger)
MsgBox(aSingle)
MsgBox(aString)
MsgBox(aBoolean)
End
```

تقوم السطور الأربعة الأولى من هذه الشفرة باستخدام الدالة function المسماة MsgBox وهي الوظيفة المسئولة عن صناديق الرسائل Message Box بإظهار قيم المتغيرات في صناديق رسائل. بينما يفيد السطر الأخير بإنهاء البرنامج.

قم بضرب المفتاح F5 من لوحة المفاتيح أو أختار Start Debugging من القائمة Debug لتشغيل البرنامج. عند تشغيل البرنامج تظهر قيم المتغيرات في صناديق الرسائل، قم بالنقر على المفتاح Ok في كل صندوق حتى يغلق الصندوق ويفتح الصندوق الذي يليه، بعد الصندوق الرابع يتم إغلاق البرنامج.

1.13.2أنواع البيانات

أنواع البيانات في Visual Basic تحدد ما هي القيم الممكن اختزانها في المتغير، وكيفية اختزانها، ونوعية العمليات الممكن إجراءها عليها.

لماذا هناك أنواع عدة من المتغيرات؟.

دع نا نف كر بهذه الطري قة، لو أن نا لدي نا ثلا ثة متغيرات، اثنان من هذه المتغيرات رقميين والثالث عبارة عن أسم، يمكن القيام بعمليات حسابية باستخدام المتغيرين الرقميين، لكن ليس باستخدام المتغير الثالث. لذلك فعندما نقوم بتعريف المتغير فنحن نحدد كيف يمكن أن يستخدم هذا المتغير.

*1.13.2.1*أنواع البيانات الرقمية

معظم برامج الحاسوب تتعامل مع الأرقام. ولما كان هناك أنواع عديدة من الأرقام فبالتالي تمتلك Visual Basic أنواع عدة من البيانات الرقمية.

تنقسم البيانات الرقمية إلى:

البيانات الرقمية الصحيحة Integral Numeric Types: و هي تلك البيانات الرقمية التي لا تحتوي على علامة عشرية، و هي بدورها تنقسم إلى عدة أنواع يبينها الجدول التالي:

النوع	مساحة الاختزان	القيمة الدنيا	القيمة العظمى
SByte	بايت (٨ بيت)	١٢٨-	١٢٧
Short	٢ بايت (١٦ بيت)	32,768-	32,767
Integer	٤ بايت (٣٢ بيت)	2,147,483,648-	2,147,483,647
Long	٨ بايت (٦٤ بيت)	9,223,372,036,854,775,808-	9,223,372,036,854,775,807
Byte	بايت (٨ بيت)	٠	٢٥٥
UShort	٢ بايت (١٦ بيت)	٠	65,535
UInteger	٤ بايت (٣٢ بيت)	٠	4,294,967,295
ULong	٨ بايت (٦٤ بيت)	٠	1.84E+18

البيانات الرقمية غير الصحيحة Non-integral Numeric Types: و هي البيانات الرقمية التي تحتوي على كسور أو علامات عشرية. وتنقسم هذه النوعية من البيانات إلى نوعين يبينهما الجدول التالي:

النوع	مساحة الاختزان	القيمة الدنيا	القيمة العظمى
Single	٤ بايت (٣٢ بيت)	±1.401298E-45	±3.4028235E+38
Double	٨ بايت (٦٤ بيت)	±1.79769313486231570E+308	±4.94065645841246544E-324

1.13.2.2أنواع البيانات اللفظية

البيانات اللفظية هي تلك البيانات التي تتكون من الحروف أو الأرقام (يشترط من استخدامها مع الأرقام، عدم استخدام هذه الأرقام في عمليات حسابية) أو العلامات الخاصة مثل & أو $، أو المسافات الفارغة.

يمكن تمثيل البيانات اللفظية بنوعين من البيانات هما:

26.النوع Char: ويستخدم لحفظ حرف واحد فقط.

27.النوع String: ويستخدم لحفظ عدد من الأحرف.

أنواع أخرى من البيانات

يستخدم Visual Basic أنواع أخرى من البيانات لتمثيل أشكال مختلفة من القيم، سنتعرف على معظم هذه الأنواع أثناء الفصول القادمة، لكن هناك بعض الأنواع التي يجدر الإشارة إليها مثل النوع Date الذي يستخدم لحفظ بيانات التاريخ والزمن، والنوع Boolean الذي يستخدم في تمثيل قيمتين أحدهما تنفي الأخرى مثل True و False.

كما يجب الإشارة إلى نوع البيانات Object الذي يتيح لك اختزان بيانات مجهولة النوع ثم تعريفها في مكان آخر من البرنامج.

1.13.3البيانات النصية

ليست كل البيانات التي يتعامل معها المبرمج من النوع الرقمي، بل يوجد نسبة معتبرة من البيانات النصية يتعامل معها المبرمج. يستخدم نوع خاص من البيانات يسمى الـ String للتعامل مع البيانات النصية. يستخدم النوع String لتمثيل سلسلة من الرموز النصية Characters مثل الحروف والأرقام والرموز الخاصة والمسافات.

يتم تعريف الـ String بنفس الطريقة التي يتم بها تعريف المتغيرات الرقمية أي باستخدام الكلمات المحجوزة Dim و As. السطر التالي يبين تعريف لمتغير نصي.

```
Dim aString As String = "This is a string"
```

عند تخصيص نص لمتغير من النوع الـ String يجب وضع النص بين علامتي تنصيص (" ").

يمكن كذلك تخصيص قيمة متغير نصير بواسطة متغير نصي آخر مثل المبين في الشفرة التالية.

```
Dim aString As String = "This is a string"
     ...
Dim bString As String = " "
bString = aString
```

في الشفرة السالفة المتغير bString يتم تخصيصه ليساوي القيمة المختزنة في المتغير aString.

يمكن استخدام علامة (&) لدمج متغيرين نصيين معاً في متغير واحد كما هو مبين في الشفرة التالية.

```
Dim aString As String = "Across the Wide"
Dim bString As String = "Missouri"
Dim cString As String = ""
cString = aString & bString
```

في المثال السابق يتم تعريف ثلاثة متغيرات من النوع String ثم يتم تخصيصها على التوالي بالقيم "Across the Wide" و " Nile" للمتغيرين الأولين، ثم يتم تعريف المتغير الثالث بأنه محصلة دمج المتغيرين الأول والثاني. هل تعلم ما هي القيمة المختزنة في المتغير الثالث؟ أنها "Across the WideNile"، نعم بلا مسافة فاصلة بينهما، وذلك لأن المتغير الأول aString لا يحتوي في نهايته على مسافة فارغة، كما لا يحتوي المتغير الثاني bString على مسافة فارغة في بدايته. لعلاج هذه المسألة يتعين علينا استخدام المسافة الفارغة " " كما هو مبين في الشفرة التالية.

```
Dim aString As String = "Across the Wide"
Dim bString As String = "Nile"
Dim cString As String = " "
cString = aString & " " & bString
```

1.13.3.1 تدريب: دمج المتغيرات النصية

الآن حان وقت التدريب. سنقوم بتجربة ما ناقشناه فيما سبق عن دمج المتغيرات النصية. أتبع الإجراءات التالية لإعداد التطبيق الذي سيتم من خلاله إختبار دمج المتغيرات.

28. من القائمة File أنقر فوق New Project.

29. من نافذة New Project أختار Template ومنها Windows Application ثم أنقر المفتاح Ok.

30. أنقر مرتين فوق النموذج ليظهر الـ Code Editor.

31. في إجراء الحدث Form1.Load قم بكتابة الشفرة التالية لتعرف أربعة متغيرات نصية وتخصصها.

```
Dim aString As String = "Concatenating"
Dim bString As String = "Without"
Dim cString As String = "With"
Dim dString As String = "Spaces"
```

قم بإضافة الشفرة التالية عقب الشفرة السابقة مباشرة حتى تستخدم الوظيفة MsgBox لإظهار نتائج دمج المتغيرات.

```
MsgBox(aString & bString & dString(
' Displays "ConcatenatingWithoutSpaces"
MsgBox(aString & " " & cString & " " & dString(
' Displays "Concatenating With Spaces"
```

قم بضرب المفتاح F5 لتنفيذ البرنامج، تلاحظ أن صندوق الرسالة الأول تعرض الرسالة ConcatenatingWithoutSpaces بينما يعرض صندوق الرسالة الثاني الرسالة Concatenating With Spaces.

1.13.4 المصفوفات Arrays

سنتعرف الآن عن المصفوفات Arrays التي تستخدم لاختزان مجموعة من القيم.

في ما سبق عرفنا أن المتغيرات تستخدم لاختزان أنواع مختلفة من البيانات ليتم استخدامها في البرامج. هناك نوع آخر من المتغيرات تسمى المصفوفة Array والتي تستخدم لاختزان قيم متعددة من نفس النوع.

لنفترض مثلاً أنك تكتب برنامج عن فريق كرة القدم وتريد اختزان أسماء اللاعبين، عندئذ قد تضطر للإعلان عن أحد عشر متغير، متغير لكل لاعب، أو الأسهل أن تعلن عن مصفوفة مستخدماً الشفرة التالية.

```
Dim players() As String
```

في هذه الشفرة قمنا بالإعلان عن مصفوفة تختزن بيانات نصية، فالقوسين () يستخدمان لإعلان المصفوفة، وعدم وجود قيمة عددية يعني أننا لا نعرف عدد القيم التي سوف تختزن في المصفوفة، أما في حالة ما إذا كنا نعرف عدد المتغيرات فيمكننا أن نعدل الشفرة لتصبح كما يلي.

```
Dim players(١٠) As String
```

لا يجب أن يخدعك العدد ١٠ الذي أضفناه بين القوسين، فلا تنسى أن الحاسوب يعد من الصفر وبذلك يكون عدد القيم التي يمكن إختزانها في هذه المصفوفة – و هو ما يسمى بطول المصفوفة – أحد عشر قيمة.

1.13.4.1 تخصيص قيم للمصفوفة

كما في أنواع البيانات الأخرى يلزمنا أن نقوم بتخصيص قيم للمصفوفة. لعمل ذلك يجب استخدام رقم يشير إلى مكان القيمة التي سوف تختزن في المصفوفة كما هو مبين فيما يلي.

```
players(0) = "John"
players(3) = "Bart"
```

في الشفرة السابقة، القيمة John تم تخصيصها لأول عنصر في المصفوفة (العنصر رقم صفر) بينما تم تخصيص القيمة Brett للعنصر الرابع للمصفوفة (العنصر رقم ثلاثة). ليس من الضروري إختزان عناصر المصفوفة بالترتيب، و في حالة عدم تخصيص قيمة للعنصر فإن قيمته تصبح هي القيمة الإفتراضية لنوع البيانات.

يمكن أيضاً الإعلان عن المصفوفة وتخصيص قيم عناصرها في سطر واحد كما يلي.

```
Dim players() As Integer = {1, 2, 3, 4, 5, 6, 7, 8, 9}
```

حيث تستخدم الأقواس من النوع {} لتخصيص القيم كما تستخدم الفاصلة (،) للفصل بين القيم. كما تلاحظ أن طول المصفوفة لم يخصص حيث أن طول المصفوفة يتم تخصيصه عن طريق عدد العناصر المحتواة بين الأقواس.

*1.13.4.2*استرجاع القيم المختزنة في المصفوفة

كما استخدمنا الأرقام لنبين المكان الذي سوف يتم اختزان القيمة فيه في المصفوفة، سوف نستخدم الأرقام مرة أخرى لتعيين القيمة المراد استرجاعها من بين قيم عناصر المصفوفة.

```
Dim AtBat As String
AtBat = players(3(
```

1.13.5التحويل بين أنواع المتغيرات

كثيراً ما يحتاج المبرمج إلى تحويل بيانات مختزنة في نوع معين من البيانات إلى نوع أخر. فيما يلي نتناول الطرق التي يمكن استخدامها لأداء مثل هذا العمل.

*1.13.5.1*تحويل المتغيرات إلى متغيرات نصية

كل المتغيرات المستخدمة في Visual Basic يمكن تحويلها إلى متغير نصي باستخدام الوظيفة CStr. في الشفرة التالية نستخدم هذه الوظيفة لتحويل قيمة من النوع Integer إلى النوع String.

```
Dim anInteger As Integer = 54
MsgBox(CStr(anInteger))
```

*1.13.5.2*التحويل بين المتغيرات الرقمية

تستخدم مجموعة من الوظائف للتحويل بين البيانات الرقمية بنفس الطريقة المستخدمة مع الوظيفة CStr. الجدول التالي يبين هذه الوظائف ونوعية البيانات التي تحول إليها.

الوظيفة	نوع البيانات التي تحول إليها
CBool(expression)	Boolean
CByte(expression)	Byte

Character	CChar(expression)
Date	CDate(expression)
Double	CDbl(expression)
Decimal	CDec(expression)
Integer	CInt(expression)
Long	CLng(expression)
Object	CObj(expression)
Signed Byte	CSByte(expression)
Short	CShort(expression)
Single	CSng(expression)
String	CStr(expression)
Unsigned Integer	CUInt(expression)
Unsigned Long	CULng(expression)
Unsigned Short	CUShort(expression)

1.14العمليات على المتغيرات

يتم استخدام المتغيرات في البرامج لاستخدامها في العمليات المختلفة. هذه العمليات قد تكون عمليات حسابية مثل الجمع والضرب و ما إلى ذلك أو قد تكون عمليات مقارنة منطقية. فيما يلي نتعلم أساسيات استخدام المتغيرات في العمليات.

1.14.1العمليات الحسابية

تعتمد العمليات الحسابية على كتابة المبرمج لتعبير Expression داخل برنامجه يقوم بإنجاز العملية الحسابية ثم يقوم بإرجاع قيمتها. التعبير هو عبارة عن قطعة من الشفرة التي تنفذ عملية حسابية ثم ترجع القيمة. كمثال تعبير الجمع البسيط المبين فيما يلي:

5+4

هذا التعبير يرجع القيمة ٩، و هو يتكون من جزأين الحدود Operands و هي القيم التي يتم إجراء العملية عليه، والعامل Operator والذي في هذه الحالة عامل الجمع (+) الذي يحدد نوع العملية الحسابية المطلوب إجراءاها.

*1.14.1.1*استخدام القيم المرتجعة من التعبيرات

حتى يصبح التعبير مفيداً في البرنامج، لابد من اختزان القيمة المرتجعة عن التعبير في متغير. الشفرة البسيطة التالية تبين طريقة لاختزان القيمة المرتجعة عن تعبير:

```
Dim anInteger As Integer = 5 + 4
```

في هذه الشفرة يتم الإعلان عن متغير anInteger من النوع الرقمي الصحيح Integer ثم يتم تخصيص القيمة المرتجعة من التعبير 5+4 باستخدام علامة التساوي (=) ليتم اختزانها في هذا المتغير.

1.14.1.2العاملات الحسابية

الاستخدام الأكثر شيوعاً للتعبيرات هو لإنجاز العمليات الحسابية مثل الجمع والطرح والضرب والقسمة. الجدول التالي يبين العاملات المستخدمة في العمليات الحسابية.

العامل	الوصف	مثال
+	يرجع حاصل جامع رقمين	٥+٤
-	يرجع الفرق بين رقمين	٥-٤
*	يرجع حاصل ضرب رقمين	٥*٤
/	يرجع خارج قسمة رقمين	٥/٤

يجب الأخذ في الاعتبار أن نوع البيانات المستخدمة في التعبير يمكنها أن تؤثر على القيمة المرتجعة من التعبير. فمثلاً قسمة رقمين ينتج عنها قيمة لا تمثل كامل الرقم الناتج. فمثلاً إذا قمنا بقسمة ٣ على ٢ فإن الناتج سوف يكون 1.5. إذا حاولت أن تخصص القيمة المرتجعة من هذا التعبير لمتغير من النوع Integer فإن الناتج سيتم تقريبه لأقرب عدد صحيح أي يكون الناتج ٢، لذلك يستحسن أن نقوم باختزان خارج القسمة على وجه الخصوص في متغير من النوع Double.

1.14.1.3تدريب: تنفيذ العمليات الحسابية

32. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

33. قم بسحب ورمي عنصر التحكم Textbox مرتين فوق النموذج.

34. قم بسحب عنصر تحكم Button ورميه فوق النموذج.

35. أنقر نقراً مزدوجاً فوق عنصر التحكم Button لتظهر نافذة محرر الشفرة Code Editor.

36.في إجراء الحدث Button1_Click قم بكتابة الشفرة التالية:

```
Dim A As Double = Textbox1.Text
Dim B As Double = Textbox2.Text

MsgBox(A + B)
MsgBox(A - B)
MsgBox(A * B)
MsgBox(A / B)
```

في السطرين الأوليين نقوم بالإعلان عن متغيرين A و B، والذي سوف نستخدمها لاختزان القيم الرقمية التي نحتاجها لإتمام عملية الجمع. ثم نستخدم خاصية Text لكل Textbox اللذان أضفناهما إلى النموذج لتلقي قيم A و B.

في السطور الأربعة الأخيرة يتم إنشاء أربعة تعبيرات لكل عمليات من العمليات الحسابية الأربعة الرئيسية، ثم نستخدم الوظيفة MsgBox لإظهار الناتج.

أضرب المفتاح F5 لتنفيذ البرنامج.

قم بكتابة رقم في كل صندوق نصي من المثبتة إلى النافذة Form1 ثم أنقر المفتاح Button1 لتنفيذ العمليات الأربعة وإظهار النتائج.

1.14.2العمليات المنطقية

في هذا الجزء نتعلم كيف نقوم بالمقارنات المنطقية باستخدام التعبيرات. سنستخدم في هذا النوع من التعبيرات ما يسمى بعاملات المقارنة Comparison Operators، هذا النوع من العاملات لا يرجع قيم عددية ولكنه يرجع قيم من النوع الـ Boolean أي True و False. تستخدم هذه العاملات بصورة متكررة في عمليات اتخاذ القرارات Decisions والتي سنتعرض لها في جزء لاحق.

الجدول التالي يبين عاملات المقارنة المختلفة.

العامل	وصفه	مثال
=	يرجع قيمة True إذا تساوي حدي المقارنة وقيمة False إذا لم يتساويا.	5 = 4 (false) 4 = 5 (false) 4 = 4 (true)
<>	يرجع قيمة True إذا لم يتساوي حدي المقارنة وقيمة False إذا تساويا.	5 <> 4 (true)

4 <> 5 (true) 4 <> 4 (false)		
5 > 4 (true) 4 > 5 (false) 4 > 4 (false)	يرجع قيمة True إذا كانت قيمة الحد الأيسر أكبر من قيمة الحد الأيمن وقيمة False إذا حدث غير ذلك.	>
5 < 4 (false) 4 < 5 (true) 4 < 4 (false)	يرجع قيمة True إذا كانت قيمة الحد الأيسر أصغر من قيمة الحد الأيمن وقيمة False إذا حدث غير ذلك.	<
5>= 4 (true) 4 >= 5 (false) 4 >= 4 (true)	يرجع قيمة True إذا كانت قيمة الحد الأيسر أكبر من أو تساوي قيمة الحد الأيمن وقيمة False إذا حدث غير ذلك.	>=
5<= 4 (false) 4 <= 5 (true) 4 <= 4 (true)	يرجع قيمة True إذا كانت قيمة الحد الأيسر أصغر من أو تساوي قيمة الحد الأيمن وقيمة False إذا حدث غير ذلك.	<=

*1.14.2.1*تدريب: استخدام عاملات المقارنة

37. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

38.قم بسحب ورمي عنصر التحكم Textbox مرتين فوق النموذج.

39.قم بسحب عنصر تحكم Button ورميه فوق النموذج.

40.أنقر نقراً مزدوجاً فوق عنصر التحكم Button لتظهر نافذة محرر الشفرة Code Editor.

41.في إجراء الحدث Button1_Click قم بكتابة الشفرة التالية:

```
Dim A As Double = CDbl(Textbox1.Text)
Dim B As Double = CDbl(Textbox2.Text)
MsgBox(A > B)
MsgBox(A < B)
MsgBox(A = B)
```

في السطرين الأولين يتم الإعلان عن متغيرين من النوع الـ Double هما A و B، حيث يتم تخصيصهما من خلال الخاصية Text لعنصر التحكم Textbox بعد تمريرهما من خلال الوظيفة CDbl لتحويلهما من النوع String – الذي يميز الخاصية Text – إلى النوع Double.

في السطور اللاحقة يتم استخدام الوظيفة MsgBox لإظهار نتيجة ثلاثة أنواع من المقارنة.

أضرب المفتاح F5 لتنفيذ البرنامج.

قم بكتابة رقم في كل صندوق نصي من المثبتة إلى النافذة Form1 ثم أنقر المفتاح Button1 لتنفيذ عمليات المقارنة وإظهار النتائج التي ستكون أما True أو False.

1.15الإجراءات Procedures

فيما يلي سنتعلم كيف ننشئ إجراء خاص بنا كيفية إنشاء بارمترات هذا الإجراء بحيث يكون هذا الإجراء يحتوي شفرة يمكن استدعاءها من مكان آخر في البرنامج.

يمكننا أن ننظر إلى الإجراء على أنه قطعة من الشفرة التي يمكنها أن تخبر البرنامج أن يقوم بفعل معين. وبالرغم من أنك قد لا تكون قادراً على استيعاب هذه الفكرة إلا أنك قد قمت باستخدام الإجراءات من خلال في التدريبات والأمثلة السابقة. فمثلاً الوظيفة MsgBox هي عبارة عن إجراء التي تقوم بفعل محدد هو إظهار الرسائل.

وبالرغم من Visual Basic تحتوي على العديد من الإجراءات النافعة، إلا انه قد تحتاج في بعض الأحيان لإجراء لا تتضمنه Visual Basic، فمثلاً تتضمن Visual Basic الوظيفة MsgBox التي تقوم بإظهار رسالة على الشاشة، لكنها لا تتضمن وظيفة لإظهار رسالة على الشاشة مصحوبة بصورة. في حال ما إذا أردت مثل هذه الوظيفة سيكون عليك إنشاء إجراء لها.

1.15.1ما هو الإجراء؟

الإجراء هو كتلة من الشفرة التي يمكن استدعاءها وتنفيذها من خلال أجزاء أخرى من الشفرة. بصورة عامة، الإجراءات تحتوي على الشفرة اللازمة لتنفيذها عملية وحيدة محددة. على سبيل المثال يمكن أن يكون لديك إجراء يسمى PlaySound يحتوي على شفرة لتشغيل ملف موسيقي من النوع wave. على حين يمكنك كتابة شفرة لتشغيل الصوت كل مرة يحتاج برنامجك إصدار صوت، من الأفضل في هذه الحالة أن تقوم بإنشاء إجراء يمكنه

عمل هذه المهمة – إصدار صوت – ثم استدعاءه بعد ذلك من أي مكان في البرنامج يتطلب أداء مهمة إصدار الصوت.

يتم تشغيل الإجراء عن طريق استدعاءها Calling في الشفرة. فمثلاً لتنفيذ الإجراء PlaySound كل ما يتطلبه الأمر إضافة سطر في المكان المخصص في البرنامج لصوت يكتب فيه أسم الإجراء كما يلي.

```
PlaySound
```

عند ما يصدل البرنا مج لهذا السطر، يقوم بالانتقال إلى الإجراء PlaySound منفذا الشفرة التي يحتويها هذا الإجراء، وعند الانتهاء من تنفيذ شفرة الإجراء ينتقل البرنامج مرة أخرى للسطر التالي للسطر الذي ذكر فيه أسم الإجراء PlaySound.

يمكنك ا ستدعاء العديد من الإجراءات كما يحلو لك. وعندئذ يتم تنفيذ الإجراءات المختلفة بذات الترتيب الذي قمت باستدعائها فيه. فمثلاً لنفترض أن لديك إجراء آخر غير PlaySound هو الإجراء DisplayResult وأنك تريد تنفيذه بعد الإجراء PlaySound، كل ما تحتاجه عندئذ هو كتابة أسم الإجراء الآخر بعد الإجراء الأول كما يلي.

```
PlaySound
DisplayResult
```

1.15.2أنواع الإجراءات

هناك نوعين من الإجراءات: الوظائف functions والروتينات الفرعية Subroutines (وهذه الأخيرة قد تدعى أحياناً Subs للاختصار). الوظيفة ترجع قيمة إلى الإجراء الذي ا ستدعاها، على حين والروتينات الفرعية تقوم بتنفيذ شفرة فقط. يتم ا ستدعاء الروتينات الفرعية بمجرد كتابة أسمها فقط كما في المثال التالي.

```
DisplayResults
```

الوظائف تختلف في عملها عن الروتينات الفرعية، فهي لا تقوم بتنفيذ الشفرات فحسب، لكنها ترجع أيضاً قيم. لنتخيل وظيفة تسمى GetDayOfWeek التي ترجع رقم صحيح يشير إلى اليوم في الأسبوع. لاستدعاء هذه الوظيفة يجب أولاً الإعلان عن المتغير الذي سوف يتم الاحتفاظ فيه بالقيمة المرتجعة، ثم نقوم بتخصيص القيمة المرتجعة لهذا المتغير في سطر آخر كما هو مبين فيما يلي.

```
Dim Today As Integer
```

```
Today = GetDayOfWeek
```

في هذا المثال، القيمة المرتجعة من الوظيفة يتم اختزانها في المتغير Today لحين استخدامها لاحقا.

1.15.3إنشاء الإجراء

عند إنشاء إجراء نقوم في البداية بالإعلان عن الإجراء. الإعلان عن الإجراء يعني العديد من الأشياء، فهو يحدد إذا كان الإجراء وظيفة أم روتين فرعي كما يحدد أسم الإجراء الذي سوف يستخدم للاستدعاء والبارمترات المطلوبة في الإجراء. الشفرة التالية تبين إعلان بسيط عن إجراء.

```
Sub MyFirstSub()
End Sub
```

الكلمة المحجوزة Sub تخبر البرنامج أن هذا الإجراء من النوع Sub وبالتالي فهو لا يرجع قيم. أسم الإجراء هو MyFirstSub بينما تشير الأقواس الفارغة ( ) إلى أن هذا الإجراء لا يحتاج بارامترات. وأخيراً تأ تي الكل مة المحجوزة End Sub لتعلن انتهاء الإجراء. الشفرة إلي سوف يقوم هذا الإجراء بتنفيذها يتم كتابتها بين هذين السطرين.

الإعلان عن الوظائف يشبه الإعلان عن الروتينات الفرعية لكن مع إضافة صغيرة هي تعريف نوع البيانات للقيم التي سوف يقوم الإجراء بإرجاعها. فمثلاً الوظيفة التي سترجع قيمة من النوع Integer تتخذ الشكل التالي.

```
Function MyFirstFunction() As Integer
End Function
```

الكلمتين المحجوزتين As Integer تشير إلى أن الوظيفة ستقوم بإرجاع قيمة من النوع Integer. لاسترجاع القيمة من الوظيفة تستخدم الكلمة المحجوزة Return كما هو مبين في المثال التالي.

```
Function GetTheNumberOne() As Integer
   Return 1
End Function
```

سوف يقوم الإجراء بإرجاع القيمة ١.

1.15.3.1 تدريب: إنشاء إجراء

42. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

43. أنقر نقراً مزدوجاً فوق النموذج حتى تظهر نافذة Code Editor.

44. قم بالبحث عن السطر الذي يحتوي على End Class و هو ما يمثل نهاية الشفرة المسئولة عن بناء النموذج. قبل هذا السطر مباشرة قم بكتابة الشفرة التالية.

```
Function GetTime() As String
 Return CStr(Now)
End Function
```

45. هذه الوظيفة تستخدم إجراء متضمن Built-in Procedure في Visual Basic تستخدم لإرجاع القيمة الحالية للوقت. ثم تستخدم الوظيفة CStr لتحويلها إلى قيمة نصية. هذه القيمة هي التي سوف ترجع من الوظيفة.

46. فوق الوظيفة التي قمت بإضافتها في الخطوة السابقة أضف ما يلي من شفرة.

```
Sub DisplayTime()
 MsgBox(GetTime)
End Sub
```

47. هذه الشفرة تمثل روتين فرعي يقوم باستدعاء الوظيفة GetTime ويعرض الناتج المرتجع عن هذه الوظيفة في صندوق رسائل.

48. أخيراً قم بإضافة هذا السطر التالي إلى الحدث Form1_Load الذي يستدعي الروتين الفرعي DisplayTime.

```
DisplayTime()
```

49. أضرب المفتاح F5 لتنفيذ البرنامج.

حينما يبدأ البرنامج، إجراء الحدث Form1_Load يتم تشغيله. هذا الإجراء يقوم باستدعاء الروتين الفرعي DisplayTime، فيقوم الروتين الفرعي باستدعاء الوظيفة GetTime والتي بدورها تقوم بإرجاع قيمة نصية تمثل الوقت الحالي للروتين الفرعي DisplayTime الذي يقوم بعرض هذه القيمة مستخدماً صندوق الرسائل.

1.15.4البارامترات

مع الوقت سنحتاج أن نمد الإجراءات ببعض البيانات. فمثلاً الإجراء PlaySound قد نحتاج أن يقوم بإ صدار صوت من ضمن عدد من الأ صوات. يمكن تحديد أي من الأصوات سوف يقوم بإصدارها الإجراء PlaySound باستخدام البارامترات.

تتشابه البارامترات مع المتغيرات. فهي لها نوع وأسم كما أنها تختزن البيانات مثلها مثل المتغيرات تماما. لكنها تختلف عن المتغيرات اختلافين أساسيين هما:

يتم الإعلان عن البارامترات داخل الإعلان عن الإجراء وليس بصورة مستقلة كما في حال المتغيرات.

البارامترات تستخدم فقط داخل الإجراءات المعلنة بداخلها.

البارامترات يتم الإعلان عنها مع الإعلان عن الإجراء داخل الأقواس التي تتبع أسم الإجراء. تستخدم الكلمة المحجوزة As لتعريف نوع بيانات البارامتر ويكون البارامتر مسبوقاً بكلمة ByVal المحجوزة، وهي الكلمة التي تضاف آلياً بواسطة Visual Basic إذا سهوت عن إضافتها.

الشفرة التالية تبين الإعلان عن البارامترات في إعلان الإجراء.

```
Sub PlaySound(ByVal SoundFile As String, ByVal Volume
As Integer)
 My.Computer.Audio.Play(SoundFile, Volume)
End Sub
```

وعند استدعاء مثل هذا الإجراء تستخدم الشفرة التالية:

```
PlaySound("Startup.wav", 1)
```

*1.15.4.1*تدريب: إنشاء وظيفة مصحوبة ببارامترات

50. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

51. قم بسحب ورمي عنصر التحكم Textbox مرتين فوق النموذج.

52. قم بسحب عنصر تحكم Button ورميه فوق النموذج.

53.أنقر نقراً مزدوجاً فوق عنصر التحكم Button1 حتى تظهر نافذة Code Editor.

54.بعد End Sub في إجراء Button1_Click قم بإضافة الشفرة التالية.

```
Function AddTwoNumbers(ByVal N1 As Integer, ByVal N2 As
Integer)
As Integer
 Return N1 + N2
End Function
```

55.في إجراء Button1_Click أضف الشفرة التالية.

```
Dim aNumber As Integer = CInt(Textbox1.Text)
Dim bNumber As Integer = CInt(Textbox2.Text)
MsgBox(AddTwoNumbers(aNumber, bNumber))
```

56.هذه الشفرة تقوم بالإعلان عن متغيرين رقميين صحيحين وتقوم بتحويل القيم النصية المرتجعة من خاصية Text إلى أرقام صحيحة. يعقب ذلك تمرير القيم الصحيحة إلى الوظيفة AddTwoNumbers ثم عرض القيمة المرتجعة في صندوق رسالة.

57.اضرب المفتاح F5 لتنفيذ البرنامج.

58. قم بكتابة رقمين ثم اضرب المفتاح Button1 لتشاهد رسالة تحمل حاصل جمع العددين.

1.16الحلقات

الحلقات هي أدوات برمجية تساعد المستخدم على أداء عمل تكراري. وتدعم لغة Visual Basic استخدام أنواع مختلفة من الحلقات. وفيما يلي نتعلم كيف يمكن استخدام كل نوع من هذه الحلقات على حدا.

1.16.1حلقة For – Next

عندما نقوم بكتابة برنامج قد نحتاج في بعض الأحيان لتكرار خطوة ما. على سبيل المثال لنفترض أنك تقوم بكتابة برنامج يقوم بعرض سلسلة من الأعداد على الشاشة. في هذه الحالة قد تضطر إلى كتابة عدد كبير من السطور التي تقوم بنفس الفعل.

تستخدم حلقة For – Next لمعالجة مثل هذه المسائل حيث تقوم بتكرار المهام عدد من المرات الذي يحدده المستخدم سلفا. المثال التالي يبين كيف يمكن استخدام حلقة For – Next.

```
Dim i As Integer = 0
For i = 1 To 10
 DisplayNumber(i)
Next
```

تبدأ حلقة For- Next بمتغير عداد counter variable هو i. هذا المتغير هو الذي تستخدمه الحلقة لعد عدد المرات المطلوب فيها تنفيذ ها. السطر التالي (For i=1 to 10) يخبر البرنامج كم من المرات عليه أن يكرر الحلقة، أي أن على البرنامج أن يقوم بتكرار الحلقة من ١ إلى ١٠. وعلى هذا يتم تكرار السطور المحصورة بين For وحتى Next من ١ إلى ١٠. وعند بلوغ عشر مرات تتوقف الحلقة عن تكرار المهام.

1.16.1.1تدريب: استخدام الحلقة For – Next

59. من قائمة File أختار New Project و من تبو يب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

60. قم بإدراج عنصري تحكم أحدهما من النوع TextBox والآخر من النوع Button على النموذج.

أنقر نقراً مزدوجاً فوق العنصر Button ليظهر Code Editor.

في الحدث Button1_Clcik أكتب الشفرة التالية:

```
Dim i As Integer = 0
Dim NumberOfRepetitions As Integer =
CInt(Textbox1.Text)
For i = 1 To NumberOfRepetitions
 MsgBox("This line has been repeated " & i & " times")
Next
```

قم بضرب المفتاح F5 لتنفيذ البرنامج.

قم بكتابة رقم في صندوق النص ثم أنقر المفتاح Button سوف يظهر صندوق النصوص عدد مساوي للرقم الذي قمت بتخصيصه في صندوق النص.

1.16.2حلقة Do-While و Do – Until

سوف نتعلم الآن كيف يمكن استخدام الحلقة Do – While ونظيرتها الحلقة Do – Until. في الجزء السابق تعلمنا كيف يمكن استخدام الحلقة For – Next لتكرار أداء مهمة عدد من المرات المحددة. لكن ماذا إذا كان العدد المطلوب تكراره من المهام غير محدد بل يعتمد على تحقق شرط معين. تقوم الحلقتين Do-While و Do-Until بالتكرار مع الأخذ في الاعتبار أن تتوقف عند تحقق شرط محدد سلفا.

كمثال لنفترض أننا نقوم بكتابة برنامج يقوم بجمع مجموعة من الأرقام، لكنك لا ترغب أن يتجاوز حاصل جمع هذه السلسلة الرقمية المائة. يمكن عندئذ استخدام حلقة Do-While كما هو مبين في الشفرة التالية.

```
Dim sum As Integer = 0
Do While sum < 100
 sum = sum + 10
Loop
```

في هذه الشفرة تقوم حلقة Do-While بتنفيذ الشفرة المحصورة بين Do While و Loop حتى يتحقق الشرط sum<100. فإذا تحقق هذا الشرط توقفت الحلقة عن التكرار.

ولا تختلف الحلقة Do Until كثيراً عن الحلقة Do While، فالحلقة Do While تظل تتكرر ما دام الشرط الخاص بها صحيحاً، أما الحلقة Do Until فهي تظل تتحقق ما دام الشرط غير صحيح فإذا تحقق الشرط توقفت الحلقة عن التكرار.

1.17القرارات

كثيراً ما يحتاج المبرمج إلى اتخاذ قرارات تتحكم في أداء البرنامج، كأن يختبر إذا ما كان مقام عملية القسمة مساوي للصفر أم لا، فإذا كان مساوياً للصفر أصدر رسالة تفيد أن العملية الحسابية غير مقبولة أو إذا كان لا يساوي الصفر أستمر في أداء العملية الحسابية.

إن مثل هذه القرارات يمكن معالجته في Visual Basic باستخدام العبارة If-Then وتوسعتها If – Then – Else والعبارة Select Case. فيما يلي سنتعلم كيف يمكن استخدام مثل هذه العبارات في اتخاذ القرارات.

1.17.1العبارة If- Then

تستخدم العبارة If – Then لاتخاذ قرار بناء على تحقق شرط ما. الشفرة التالية تبين طريقة استخدام If – Then.

```
Dim A As Integer = 7
Dim B As Integer = 4
If (A>B) Then
     MsgBox ("The Condition is true.")
End If
```

في الشفرة السالفة نقوم بتعريف متغيرين من النوع الحقيقي هما A و B ونقوم بتخصيص قيمة لكل منهما على التوالي. ثم نستخدم العبارة If لاختبار تحقق الشرط A>B إذا تحقق هذا الشرط تظهر رسالة تفيد بتحقق الشرط و هي الرسالة التي يظهرها صندوق الرسالة بين Then و End If.

إن قاعدة استخدام If – Then بسيطة للغاية، إذا تحقق الشرط إذا If نفذ Then مجموعة العبارات اللاحقة حتى End If.

1.17.2التوسعة If-Then-Else

يمكن استخدام العبارة If – Then في صيغتها الموسعة بإضافة Else للتخيير بين قرارين بناء على شرط. الشفرة التالية تعتمد على الشفرة السابقة بعد إضافة لها التوسعة Else لها.

```
Dim A As Integer = 7
Dim B As Integer = 4
If (A>B) Then
     MsgBox ("The Condition is true.")
Else
     MsgBox ("The Condition is false.")
End If
```

في هذه الحالة تصبح الصيغة لهذه الشفرة هي كالتالي: إذا If تحقق الشرط (A>B) إذاً Then بطباعة الرسالة The condition is true وإلا Else أطبع العبارة The condition is false حيث لم يتحقق الشرط ثم قم بإنهاء العبارة End If.

1.17.2.1 تدريب: مقارنة عددين

61. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

62. قم بإدراج عنصري تحكم أحدهما من النوع TextBox والآخر من النوع Button على النموذج.

63. أنقر نقراً مزدوجاً فوق العنصر Button ليظهر Code Editor.

64. في الحدث Button1_Clcik أكتب الشفرة التالية:

```
Dim A As Integer = CInt(TextBox1.Text)
Dim B As Integer = CInt(TextBox2.Text)
If (A > B) Then
   MsgBox(A & " is the bigger number")
Else
   MsgBox(B & " is the bigger number")
End If
```

65. قم بضرب المفتاح F5 لتشغيل البرنامج.

66. قم بتخصيص قيمتين رقميتين في صندوقي النصوص ثم أنقر المفتاح Button1 لتنفيذ عملية المقارنة.

1.17.3 العبارة Select Case

عندما يكون عدد القرارات يزيد عن قرارين يصبح استخدام العبارة If – Then مجهداً. لذلك تقدم Visual Basic خياراً آخر للتعامل مع القرارات المتعددة هو العبارة Select Case.

تستخدم العبارة Select Case لاختيار قرار من بين عدة قرارات بناء على شرط منطقي. الشفرة التالية تبين كيف يمكن استخدام العبارة Select Case.

```
Dim A As Integer = 1
Select Case A
Case 1
      MsgBox("White")
Case 2
      MsgBox("Black")
Case 3
      MsgBox("Red")
End Select
```

يمكن قراءة هذه الشفرة كما يلي.

أختر على أساس Select Case قيمة A فإذا كانت (Case 1 (A=1 فقم بإظهار الرسالة White أو إذا كانت قيمة (Case 2 (A=2 فقم بإظهار الرسالة Black أو إذا كانت قيمة (Case 3 (A=3 فقم بإظهار الرسالة Red. ثم قم بإنهاء العبارة End Select.

1.17.4التوسعة Select Case – Case Else

تعمل التوسعة Case Else مع العبارة Select Case عمل يشابه عمل Else مع عبارة If فهي تقدم البديل في حالة عدم تحقق أي من الشروط التي تختبر ها العبارة Select Case. الشفرة التالية تقدم فكرة أوضح عن استخدام التوسعة Case Else.

```
Dim A As Integer = 1
Select Case A
Case 1
      MsgBox("White")
Case 2
      MsgBox("Black")
Case 3
      MsgBox("Red")
Case Else
      MsgBox("Please choose 1,2 or 3")
End Select
```

في هذه الشفرة تصبح الإضافة هي كالآتي، في حال لم يتحقق اي من الشروط السابقة Case Else قم بطباعة العبارة Please choose 1,2 or 3.

1.17.4.1 تدريب: الاختيار من بين عدة قيم

67. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

68. قم بإدراج عنصري تحكم أحدهما من النوع TextBox والآخر من النوع Button على النموذج.

69. أنقر نقراً مزدوجاً فوق العنصر Button ليظهر Code Editor.

70. في الحدث Button1_Clcik أكتب الشفرة التالية:

```
Select Case A
        Case 1
          MsgBox("White")
        Case 2
          MsgBox("Black")
        Case 3
          MsgBox("Red")
        Case Else
          MsgBox("Please choose 1,2 or 3")
End Select
```

71. قم بضرب المفتاح F5 لتشغيل البرنامج.

72. قم بتخصيص قيمة رقمية في صندوق النصوص ثم أنقر المفتاح Button1 لتنفيذ عملية المقارنة.

1.18 معالجة الأخطاء

سنقوم الآن بتعلم كيف يمكننا معالجة الأخطاء في شفرة البرنامج. حتى أفضل البرامج أحيانا ما ينتج عنها أخطاء. بعض هذه الأخطاء هي عبارة عن عيوب في شفرة البرنامج يمكن اكتشافها وتحديدها. بعضها الآخر غير ناتج عن البرنامج مثلاً ولكن عن بيئة التشغيل، كأن يحاول البرنامج فتح ملف مفتوح في برنامج آخر. في مثل هذه الحالات يمكن التنبؤ بمثل هذه الأخطاء وللكن لا يمكن منعها. وكمبرمج فمن ضمن واجباتك التنبؤ بمثل هذه الأخطاء وتكيف برنامجك للتعامل معها.

1.18.1الأخطاء من النوع Run Time

يطلق على الخطأ الذي يحدث أثناء تشغيل البرنامج الاسم Run Time Error. هذا النوع من الأخطاء يحدث عندما يحاول البرنامج أن يقوم بعمل لم يصمم البرنامج من أجله. مثلاً أن يقوم البرنامج بعملية غير مقبولة Illegal Operation كأن يقوم بتحويل نص غير رقمي إلى متغير رقمي، في مثل هذه الحالات تظهر أخطاء الـ Run Time.

عند ما يـ قع هذا النوع من الأخطاء أثناء التطو ير في بيئة Visual Basic ينشئ Visual Basic ما يسمى بالاستثناء Exception و هو ما يسمح للمبرمج أن يتعامل مع الخطأ عن طريق النظر إلى الشفرة داخل البرنامج ومعالجة الخطأ. إذا لم توجد شفرة معيبة فإن البرنا مج سوف يتو قف وعندئذ ستحتاج إلى إعادة تشغيله مره أخرى. و في مـ ثل هذه الحالات غالباً ما يحدث فقد في البيانات التي يدير ها البرنامج لذلك يفضل البحث عن هذه الأخطاء أياً كان موضعها وإزالتها.

1.18.2العبارة Try – Catch – Finally

يمكنك استخدام العبارة Try – Catch – Finally لمعالجة الأخطاء من النوع Run Time في شفرتك. منطق هذه العبارة بسيط للغاية، فيمكنك أن تجرب Try جزء من الشفرة، فإذا حدث أي استثناء في الشفرة، ينتقل البرنامج إلى السطور التالية لـ Catch ويقوم بتنفيذ هذه السطور. وبعد الانتهاء من تنفيذ هذه الشفرة فإن أي شفرة بعد Finally لابد من تنفيذها. الشفرة التالية تبين منطق هذه العبارة.

```
Try
  ' Code here attempts to do something.
Catch
  ' If an error occurs, code here will be run.
Finally
  ' Code in this block will always be run.
End Try
```

بدا ية الشفرة اللاحقة لـ Try يتم تنفيذ ها. إذا نفذت هذه الشفرة دون أخطاء يقفز البرنامج فوق Catch وما يلحقها من شفرة ويتم تنفيذ الشفرة اللاحقة لـ Finally. إذا حدث خطأ في الشفرة اللاحقة لـ Try ينتقل البرنامج للشفرة اللاحقة لـ Catch ثم يقوم بتنفيذ الشفرة اللاحقة لـ Finally.

1.18.2.1 تدريب: استخدام *Try- Catch- Finally*

73. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

74. قم بإدراج عنصري تحكم أحدهما من النوع TextBox والآخر من النوع Button على النموذج.

75. أنقر نقراً مزدوجاً فوق العنصر Button ليظهر Code Editor.

76. في الحدث Button1_Clcik أكتب الشفرة التالية:

```
Try
 Dim aNumber As Double = CDbl(Textbox1.Text)
 MsgBox("You entered the number " & aNumber)
Catch
 MsgBox("Please enter a number.")
Finally
 MsgBox("Why not try it again?")
End Try
```

قم بضرب المفتاح F5 لتشغيل البرنامج.

قم بتخصيص قيمة رقمية في صندوق النصوص ثم أنقر المفتاح Button1 لتنفيذ عملية الاختبار.

بناء واجهات التشغيل

واجهة التشغيل User Interface هي ذلك الجزء من البرنامج الذي سوف يراه مستخدم البرنامج عند تشغيله. عادة ما تتكون واجهة التشغيل من نافذة رئيسية أو نموذج وعدد من عناصر التحكم مثل المفاتيح وخانات إدخال النصوص و ما إلى ذلك. البرامج المطورة بـ Visual Basic التي تعمل على حاسوبك يطلق عليها أسم تطبيقات نماذج النوافذ Windows Forms Applications ويستخدم لبناء واجهات تشغيلها عناصر Windows Forms.

في هذا الفصل نتعلم كيف نقوم ببناء واجهة التشغيل باستخدام أكثر عناصر تحكم Windows Forms استخداما.

1.19واجهة التشغيل User Interface

سنتعلم هنا ما هي واجهة التشغيل User Interface (UI) وما هي عناصر التحكم Controls وكيف يمكن إضافة عنصر التحكم إلى واجهة التشغيل.

في بدايات الحواسيب، كان مستخدمي البرامج يتعاملون مع برامجهم من خلال سطر الأوامر Command Line. كان البرنامج يبدأ ثم ينتظر تلقي البيانات من المستخدم قبل أن يستمر. معظم البرامج التي تستخدمها اليوم تشتغل من خلال نافذة أو أكثر لتسمح لمستخدمها بأن يتعامل مع البرنامج من خلال الضرب على لوحة المفاتيح أو بالنقر على المفاتيح أو الاختيار من القوائم وما إلى ذلك. سنقوم الأن برحلة الغرض منها تعلم كيف نقوم ببناء واجهة تشغيل لبرامجنا تعمل من خلال النوافذ.

1.19.1بناء النماذج Forms

النماذج Forms هي البنى الأساسية لواجهة التشغيل. كل نموذج في البرنامج يظهر للمستخدم في صورة نافذة Window. عند العمل في بيئة التطوير المتكاملة لـ Visual Basic (Visual Basic Integrated Development Environment IDE)، يمثل النموذج لوحة التصميم التي تستخدمها لتصميم واجهة تشغيل برنامجك، بنفس الطريقة التي تستخدم برنامج Windows Paint لرسم الأشكال.

عناصر التحكم تستخدم فوق لوحة التصميم هذه لبناء مظهر الواجهة. عنصر التحكم ينظر إليه ككائن محدد سلفاً المظهر والسلوك. مثال على ذلك المفتاح Button له مظهر وسلوك معينين، فعند النقر عليه يتغير شكله ليبين أن نقر.

لكل عنصر تحكم تقدمه Visual Basic وظيفة محددة. مثلاً عنصر التحكم TextBox يستخدم لإدخال النصوص، أما عنصر التحكم PictureBox فيستخدم لعرض الصور. تقدم Visual Basic لمستخدميها نحو خمسين عنصر تحكم مختلف، إضافة إلى أن المبرمج يمكنه أن يقوم بتصميم عناصر تحكمه الخاصة والتي يطلق عليها عناصر تحكم المستخدم User Controls.

عند تصميم واجهة التشغيل، كل ما تحتاجه هو أن تقوم بسحب Drag عنصر التحكم من الـ Toolbox، ثم رميه Drop فوق المكان المناسب على النموذج، بعد ذلك يمكنك أن تغير موضعه وحجمه فوق النموذج بما يلاءم تصورك عن واجهة تشغيل برنامجك. كما يمكنك تغيير مظهر عناصر التحكم عن طريق تغيير خصائصها في نافذة الخصائص Properties. تمنح نافذة الخصائص العديد من الخصائص القيمة التي تساعد المبرمج على تحوير مظهر عناصر التحكم، فمثلاً النماذج وعناصر التحكم يكون لها لون خلفية يمكن ضبطه من الخاصية المسماة BackColor.

يمكن أيضاً استخدام الخصائص لتعريف سلوك النموذج أو عنصر التحكم. مثلا الخاصية ShowInTaskbar للنموذج تحدد ما إذا كان النموذج سوف يظهر في شريط مهام Windows أم لا عند تشغيل البرنامج.

1.19.1.1 *تدريب: تغيير خصائص النموذج*

77. من قائمة File أختار New Project ومن تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

78. أنقر فوق النموذج لتختاره موضوع عملك.

79. في نافذة Properties قم بتغيير خاصية Text لتصبح My First Form ثم أضرب مفتاح Enter في لوحة المفاتيح، هل تلاحظ تغير النص المكتوب في شريط عنوان Title Bar النموذج من Form1 إلى My First Form.

80.في نافذة الخصائص قم بتغيير خاصية BackColor إلى لون آخر باختيار لو من القائمة المنسدلة ليتغير لون خلفية النموذج.

1.19.1.2 تدريب: إضافة عناصر التحكم إلى النموذج

سنتابع في هذا التدريب بناء واجهة تشغيل فوق النموذج الذي أعددناه في التدريب السالف.

81.من Toolbox قم بسحب عنصر التحكم Button ورميه فوق النموذج ثم كرر العمل مع عناصر التحكم TextBox و Label و CheckBox.

82.أنقر فوق الــ Button لتختاره ثم قم بسحبه فوق النموذج مغيراً من موقعه.

83.كرر العملية لباقي عناصر التحكم.

84.أنقر فوق الــ Button لتختاره ثم أنقر وأسحب المربع الأبيض الصغير المثبت إلى الحافة السفلية للمفتاح لتغيير حجمه.

85.حاول أن تختبر خصائص عناصر التحكم الأخرى مثل Font و BackColor و ForeColor و Text.

86.قم بضرب المفتاح F5 لتشغيل البرنامج. هل تلاحظ الهيئة الجديدة للبرنامج.

1.20 التفاعل مع المستخدم: استخدام المفاتيح

فيما يلي نتعلم كيف نضيف عنصر التحكم Button إلى النموذج وكيف نقوم بتغيير مظهره وكيف نكتب شفرة يتم تنفيذها عندما ننقر عليه.

تعتبر أسهل وسيلة لكي يتفاعل المستخدم مع البرنامج هي المفاتيح Buttons. فمثلا كثير من البرامج تحتوي مفتاح Exit لإنهاء البرنامج. كما رأيت سلفاً يأخذ المفتاح مظهر وسلوك المفاتيح العادية المثبتة إلى الأجهزة مثلا. بالإضافة إلى المظهر والسلوك للمفاتيح مجموعة من الأحداث Events المعرفة سلفاً والتي تستخدم لتنفيذ الشفرات.

1.20.1استخدام المفاتيح

المفاتيح بصورة عامة لها مظهر مستطيل بارز فوق النماذج. لكن هذا المظهر يمكن تغييره باستخدام العديد من الخصائص. أوضح هذه الخصائص هي خاصية Text والتي تحدد النص الذي سوف يعرض على المفتاح، وهذا النص سيعرض بخط يتم التحكم فيه من خاصية Font. خاصية BackColor تحدد لون المفتاح بينما تحدد خاصية ForeColor لون النص فوق المفتاح.

عندما ينقر مستخدم البرنامج مفتاحاً، فإن المفتاح يبدأ تشغيل شفرة الحدث Click. عند وقوع حدث Event فإن الشفرة المرفقة في هذا الحدث يبدأ تنفيذها. يمكنك كتابة الشفرة التي سوف يجرى تنفيذها بواسطة بناء عامل الحدث event handler.

عامل الحدث Event handler هو طريقة Method يتم تنفيذها عند وقوع الحدث. حينما ينقر المستخدم فوق المفتاح، حدث Click للمفتاح يكون له event handler. وفي موضع لاحق سنتعلم كيف نتعامل مع عامل الحدث Event handler.

1.20.1.1تدريب: استخدام المفتاح

87. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

88.قم بسحب Button من Toolbox ورميه فوق الـ Form.

89.في نافذة الخصائص قم بتغيير خاصية Text لتصبح What time is it؟ ثم أضرب مفتاح Enter من لوحة المفاتيح.

90.هل لاحظت أن النص أكبر من المفتاح، أختار الخاصية AutoSize ثم أضبط قيمتها لتصبح True، يتغير حجم المفتاح ليناسب حجم النص.

91.أنقر مرتين فوق المفتاح لتظهر نافذة Code Editor.

92.في عامل الحدث Event handler أكتب الشفرة التالية.

```
MsgBox("The current time is " & Now.ToShortTimeString)
```

93.أضرب المفتاح F5 ليتم تشغيل البرنامج، أنقر المفتاح لتظهر لك رسالة تحمل التوقيت الآن.

1.21استخدام النصوص

سوف نتعلم الآن كيفية استخدام عنصري التحكم Label و TextBox لعرض النصوص وإدخال البيانات النصية من مستخدم البرنامج.

واحدة من أسهل الطرق لاستلام وعرض البيانات في البرنامج هي من خلال النصوص. حيث يمكنك عرض بيانات حول وظائف عناصر واجهة برنامجك، أو تسلم البيانات من المستخدم لاستخدامها في برنامجك. تقدم Visual Basic عنصري التحكم Label و TextBox لأداء هذه المهام.

1.21.1عرض النصوص في عنصر التحكم Label

عنصر التحكم Label هو عنصر تحكم رئيسي لعرض النصوص. يظهر عنصر التحكم Label فوق النموذج أثناء التصميم كنص محاط بإطار مستطيل. يكون لون عنصر التحكم Label هو نفس لون النموذج لذلك يظهر أثناء تشغيل البرنامج مجرد نص فوق النموذج.

حيث أن وظيفة عنصر التحكم Label الأساسية هي عرض النص، فإن الخصائص الأهم لعنصر التحكم Label هي الخصائص التي تحدد مظهره. خاصية Text هي المسئولة عن النص المعروض في عنصر التحكم Label. خاصية Font مسئولة عن تحديد مظهر الخط المستخدم في إظهار النص في عنصر التحكم Label. الخاصية ForeColor يحدد لون النص نفسه، أما الخاصية BackColor فتحدد لون المنطقة المحيطة بالنص.

1.21.2تسلم النص عبر عنصر التحكم TextBox

في حالة ما إذا رغبت في عرض واستلام النص، يطرح Visual Basic عنصر التحكم TextBox لأداء هذه المهمة. كما في حالة عنصر التحكم Label، الخصائص الأهم بالنسبة لعنصر التحكم TextBox هي تلك المرتبطة بمظهره. الخاصية Text تحدد النص المعروض في عنصر التحكم. الخاصية Multiline تحدد ما إذا كان عنصر التحكم TextBox قابل لتسلم نصوص وعرضها على أكثر من سطر، حيث تأخذ هذه الخاصية

القيمة False لعرض وتسلم سطر واحد من البيانات، أو القيمة True لعرض وتسلم أكثر من سطر من البيانات.

1.21.2.1 تدريب: استخدام عنصري التحكم Label و TextBox

94. من قائمة File أختار New Project و من تبو يب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

95. قم بسحب عناصر التحكم Label و TextBox و Button من Toolbox و رميهم فوق النموذج.

96. أختر عنصر التحكم Label ثم أذهب إلى نافذة Properties و قم بتغيير قيمة الخاصية Text إلى Enter your name and click the button.

97. أنقر نقراً مزدوجاً فوق عنصر التحكم Button لتظهر نافذة Code Editor.

98. قم بكتابة الشفرة التالية في عامل الحدث Button1_Click

```
MsgBox("Your Name is " & Textbox1.Text)
```

أضرب المفتاح F5 لتشغيل البرنامج.

قم بكتابة أسمك في خانة النص ثم أنقر المفتاح Button لتظهر رسالة نص تحتوي على أسمك.

1.22 بناء عامل حدث Event Handler

سوف نتعلم الآن كيفية بناء عامل حدث Event Handler.

فيما سبق تعرفنا أن عناصر التحكم لها خصائص Properties وطرق Methods وأحداث Events، وتستخدم عناصر التحكم هذه لإنشاء واجهة التشغيل لبرامجك.

عندما يحدث شئ متوقع بالنسبة لعنصر التحكم – نقر على مفتاح مثلاً – يقوم عنصر التحكم بإطلاق الحدث Event، بمعنى أنه يقوم بإرسال إشارة إلى البرنامج تعلمه بأن هناك شئ مهم بالنسبة له قد حدث. يقوم البرنامج بعد ذلك باستكشاف ما إذا كان هناك أي طرق معرفة للعمل عند وقوع هذا الحدث. مثل هذه الطرق تسمى عامل الحدث Event Handler.

يمكنك إنشاء عوامل الأحداث للعديد من أحداث عناصر التحكم. فيما يلي سنقوم ببناء عوامل أحداث لمعالجة أحداث MouseEnter و MouseLeave الخاصة بعنصر التحكم Button، وهي الأحداث التي تقع عند مرور الفأرة فوق عنصر التحكم.

1.22.1تدريب: معالجة حدث MouseEnter

99. من قائمة File أختار New Project و من تبو يب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

قم بسحب عنصر تحكم Button من Toolbox ورميه فوق النموذج.

قم بتغيير قيمة خاصية AutoSize لعنصر التحكم لتصبح True.

من قائمة View أختار Code، تظهر نافذة Code Editor لاحظ الصندوقين المنسدلين في أعلى نافذة Code Editor. الصندوق الأيسر يضم كافة عناصر التحكم المستخدمة في تصميم الواجهة. والصندوق الأيمن يضم كافة الأحداث المتاحة لعنصر التحكم المختار من الصندوق الأيمن.

في الصندوق الأيمن أختار Button1.

في الصد ندوق الأي من أختار الحدث MouseEnter يظ هر عا مل الحدث Event Handler المسمى Button1_MouseEnter في نافذة Code Editor.

قم بكتابة الشفرة التالية في الحدث.

```
Button1.Text = "The Mouse has entered"
```

قم بضرب المفتاح F5 من لوحة المفاتيح، قم الأن بتمرير الفأرة فوق المفتاح ولاحظ تغير نص المفتاح.

1.22.2إضافة عامل حدث آخر

لعلك لاحظت في المثال السابق أن النص الأصلي للمفتاح لا يعود كما كان عند ما تنزاح الفأرة من فوقه. إذا كنت ترغب في تغيير النص بمجرد انزياح الفأرة لابد من إضافة عامل حدث آخر هو عامل الحدث MouseLeave.

1.22.2.1تدريب: إضافة الحدث *MouseLeave*

100.من صندوق الأحداث إلى اليسار أختار الحدث MouseLeave.

101.في عامل الحدث Button1_MouseLeave أضف الشفرة التالية:

```
Button1.Text = "The mouse has left"
```

102.أضرب المفتاح F5 وجرب البرنامج.

1.22.3مشاركة عامل الحدث

فيما يلي، سوف نتعلم كيف نقوم عامل حدث مشترك يتعامل مع أحداث لأكثر من عنصر تحكم. فيما سبق تعلمنا كيف نكتب شفرة تستجيب لحدث MouseEnter وحدث MouseLeave لعنصر التحكم Button. ماذا يحدث إذا كان لدينا أكثر من عنصر تحكم Button وأنت تريد أن تعرض نفس الرسالة لجميع هذه العناصر؟. في هذه الحالة يمكن كتابة نفس شفرة عامل الحدث لكل عنصر من عناصر التحكم، لكن ولحسن الحظ تقدم Visual Basic خيار أفضل.

لو كنت تفحصت مجموعة طرق عوامل الأحداث للحدث MouseEnter كنت ستلاحظ أن تعريف الطريقة

```
Private Sub Button1_MouseEnter(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Button1.MouseEnter
```

يحتوي على جزء هام هو

```
Handles Button1.MouseEnter
```

إن الكلمة المحجوزة Handles تفيد عامل الحدث عن أي الأحداث التي يجب عليه أن يقوم بمعالجتها.

لمشاركة عامل حدث بين أكثر من عنصر تحكم، فإنه يلزمك أن تضيف أسماء عناصر تحكم إضافية وأسم الحدث الذي ترغب في معالجته. فمثلاً إذا كان هناك عنصري تحكم Button وترغب في أن تستخدم نفس عامل الحدث لهما، فإن الكلمة المحجوزة Handles سوف تظهر في الشفرة كما يلي:

```
Handles Button1.MouseEnter, Button2.MouseEnter.
```

الآن لديك طريقة Method واحدة لمعالجة الحدث MouseEnter لكلا عنصري التحكم. لكن كيف يمكن لعامل الحدث أن يعرف أي عنصري التحكم هو من أطلق الحدث؟. إذا ألقيت مرة أخرى النظر على إعلان Method سوف تلاحظ الكلمة المحجوزة ByVal sender As Object. إن الكلامة المحجوزة Sender تفيد عامل الحدث عن الكائن – عنصر التحكم – الذي أطلق الحدث.

1.22.3.1 تدريب: المشاركة في عامل حدث

هذا التدريب يعتمد على التدريب السابق حيث سنقوم بتعديل الواجهة والشفرة الخاصة بعناصر التحكم.

103.من Toolbox قم بسحب Button جديد وإضافته إلى الـ Form.

104.قم بالنقر مرتين فوق Button1 لفتح نافذة Code Editor.

105. في الحدث MouseEnter قم بتعديل هذا الجزء من الإعلان عن الطريقة Handles Buuton1.MouseEnter ليصبح Handles Button1.MouseEnter, Button2.MouseEnter.

106. قم بحذف الشفرة الموجودة داخل الطريقة Method لعامل الحدث Button1_MouseEnter واستبدالها بالشفرة التالية.

```
If sender.Equals(Button1) Then
 Button1.Text = "The mouse has entered Button1"
Else
 Button2.Text = "The mouse has entered Button2"
End If
```

107.الآن قم بضرب المفتاح F5 وأختبر البرنامج.

1.23 استخدام مربعات التأشير ومفاتيح الراديو

فيما يلي سنتعلم كيفيه استخدام مربعات التأشير Check Boxes ومفاتيح الراديو Radio Buttons، وهي عناصر التحكم المسئولة عن تلقي اختيارات مستخدمي البرامج.

1.23.1مربع التأشير Check Box

عندما تقوم بإنشاء واجهة تشغيل لبرنامجك، عادة ما تحتاج أن تقدم اختيارات. مثلاً، أفترض أنك تقوم بكتابة برنامج لتلقي طلبات مطعم بيتزا، ربما كنت تريد أن تجعل مستخدم برنامجك أن يختار من بين عدد من أصناف البيتزا. عنصر التحكم CheckBox يقدم أداة جيدة يمكن للمستخدم أن يستخدمها في تحديد خياراته.

يتكون عنصر التحكم CheckBox من نص عنوان ومربع حيث يمكن للمستخدم أن يضع إشارة الموافقة. عندما ينقر المستخدم فوق مربع التأشير تظهر إشارة الموافقة في المربع. إذا تم التأشير على المربع مرة ثانية فإن إشارة الموافقة تختفي. إن حالة مربع التأشير من حيث هو يحمل إشارة الموافقة أو لا يمكن التعرف عليها من خلال الخاصية CheckBox.Checked. لو أن المربع يحوي الإشارة فإن هذه الخاصية تصبح True أما إذا كان لا يحمل هذه العلامة فإن قيمة الخاصية تصبح False.

*1.23.1.1*تدريب: استخدام مربع التأشير

108. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

قم بسحب عنصر التحكم CheckBox من ToolBox ورميه فوق النموذج، وكرر هذا الإجراء مرتين إضافيتين.

قم بسحب عنصر تحكم Button من Toolbox ورميه فوق النموذج.

أختار عنصر التحكم CheckBox1 من فوق النموذج ثم اذهب إلى نافذة الخصائص وعدل الخاصية Text لتصبح Pepperoni، ثم كرر الإجراء لعنصر التحكم CheckBox2 جاعلاً خاصية Text له هي Sausage ونفس الأمر لعنصر التحكم الذي تصبح خاصية Text له هي Mushrooms.

في خصائص Button1 أجعل الخاصية Text هي Order Pizza.

أنقر نقراً مزدوجاً فوق عنصر التحكم Button1 لتظهر نافذة Code Editor ثم قم بكتابة الشفرة التالية:

```
Dim toppings As String = ""
If CheckBox1.Checked = True Then
   toppings &= "Pepperoni "
End If
If CheckBox2.Checked = True Then
   toppings &= "Sausage "
End If
If CheckBox3.Checked = True Then
   toppings &= "Mushrooms"
End If
If toppings <> "" Then
   MsgBox("Your pizza has the following toppings: " & 
toppings(
End If
```

قم بضرب F5 ثم أختار أحد الخيارات الثلاثة وأنقر المفتاح Order Pizza.

1.23.2مفتاح الراديو Radio Button

تعلمنا كيف يمكن أن نقوم بالسماح لمستخدمي برامجنا بتحديد إختيارتهم. لكن ماذا إذا كان علينا أن نجعل المستخدم يقوم بإختيار واحد فقط من ضمن مجموعة خيارات؟. في هذه الحالة فإن عنصر التحكم المثالي لهذه الوظيفة هو عنصر التحكم RadioButton.

وعلى عكس مربعات التأشير غالباً ما تعمل مفاتيح الراديو (والتي يطلق عليها أحيانا أسم مفاتيح الخيارات Option Buttons) كمجموعة. في هذه المجموعات يكون اختيار أي منها بمثابة عدم اختيار باقي الخيارات.

يمكن مثلاً أن تستخدم مجموعة من مفاتيح الراديو لتسمح لمستخدم برنامج طلبات البيتزا بأن يحدد ما إذا كان يريد صوص عادي أو صوص حار فوق البيتزا. وكما في حالة مربع التأشير يمكن التعرف على ما إذا كان مفتاح الراديو يحمل علامة الموافقة أم لا من خلال الخاصية RadioButton.Checked.

*1.23.2.1*تدريب: إضافة مفاتيح الراديو

استكمالا على التطبيق السابق، نقوم بتنفيذ الإجراءات التالية:

109.قم بسحب عنصر التحكم RadioButton من Toolbox ورميه فوق الـ Form مرتين.

110. من نافذة الخصائص، أختار خاصية Text لعنصر التحكم RadioButton1 وأجعلها Regular Sauce.

من نافذة الخصائص، أختار خاصية Text لعنصر التحكم RadioButton2 وأجعلها Hot Sauce.

قم بالنقر مرتين فوق عنصر التحكم Button1 لتظهر نافذة Code Editor.

أضف هذه الشفرة أسفل الشفرة التي قمنا بكتابتها التدريب السابق.

```
If RadioButton1.Checked = True Then
   MsgBox("You chose regular sauce")
Else
   MsgBox("You chose spicy sauce")
End If
```

قم بضرب المفتاح F5 لتشغيل واختبار البرنامج.

1.23.3استخدام أكثر من مجموعة من مفاتيح الراديو

سنتعلم الآن كيف ننشئ عدد من مجموعات مفاتيح الراديو فوق نفس النموذج. ففيما سبق تعلمنا أن ننشئ مجموعة من مفاتيح الراديو التي تقدم مجموعة من الخيارات للمستخدم. ماذا يحدث عند ما نرغب في تقديم مجموعتين أو أكثر من الخيارات؟. إذا أضفنا مفاتيح الراديو التي تمثل مجموعات الخيارات كلها فوق نفس النموذج سنجد أنها تتصرف كأنها مجموعة واحدة، أي أن اختيار أي منها يعني عدم اختيار الباقين.

يقدم Visual Basic مجموعة من عناصر التحكم التي يطلق عليها أسم الحاويات Containers والتي تعمل بمثابة حاوية لعناصر التحكم. بإضافة أي من الحاويات إلى النموذج ووضع مجموعة من مفاتيح الراديو بداخلها، فإن هذه المجموعة تصبح مستقلة عن غيرها من مفاتيح الراديو الخارجة عن الحاوية.

الحاوية الأكثر شيوعاً هي عنصر التحكم GroupBox وعنصر الحكم Panel. الفارق الرئيسي بينهما هو أن عنصر التحكم GroupBox له حافة مرئية حوله، ولا يوجد

مثل هذه الحافة حول عنصر التحكم Panel. عند استخدام حاوية لتجمع مجموعة من مفاتيح الراديو، فإن عنصر التحكم GroupBox هو الخيار الأفضل، لأن حافته تبين لمستخدم البرنامج أن هذه المجموعة من المفاتيح التي يحتويها هي مجموعة واحدة لا يمكنه إلا أن يختار إحداها فقط.

1.23.3.1 تدريب: استخدام عنصر التحكم *GroupBox* كحاوية

استكمالا للمشروع السابق نقوم بالإجراءات التالية:

111. من Toolbox قم بسحب GroupBox ورميه فوق النموذج.
112. من نافذة الخصائص قم بتغيير خاصية Text لعنصر التحكم GroupBox لتصبح Select a crust.
113. أختر الـ GroupBox فوق النموذج ثم قم بسحب RadioButton من Toolbox ورميه فوق الـ GroupBox، وكرر هذه الخطوة ليصبح عندنا مفتاحي راديو فوق الحاوية.
114. في نافذة الخصائص قم بتغيير خاصية Text لعنصري التحكم RadioButton1 و RadioButton2 على التوالي لتصبح Thin crust و Thick crust.
115. أنقر مرتين فوق المفتاح Order Pizza لتظهر نافذة Code Editor. أضف هذه الشفرة أسفل الشفرات الموجودة في عامل الحدث Button1_Click.

```
If RadioButton3.Checked = True Then
  MsgBox("You chose a thin crust")
Else
  MsgBox("You chose a thick crust")
End If
```

116. أضرب المفتاح F5 وقم بتجربة البرنامج.

1.24 استخدام الصور

فيما يلي نتعلم كيفية استخدام عنصر التحكم PictureBox لعرض الصور واستخدام الصورة كخلفية للنموذج.

1.24.1عنصر التحكم PictureBox

هناك عدة طرق لعرض الصور في Visual Basic أهمها هو عنصر التحكم PictureBox. يعمل عنصر التحكم PictureBox بمثابة حاوية للصور، ويمكنك أن تختار الصورة التي سوف يتم عرضها فيها من خلال الخاصية Image. هذه الخاصية يمكن ضبطها من خلال نافذة الخصائص أو يمكنك كتابة شفرة لتحميلها أثناء عمل البرنامج.

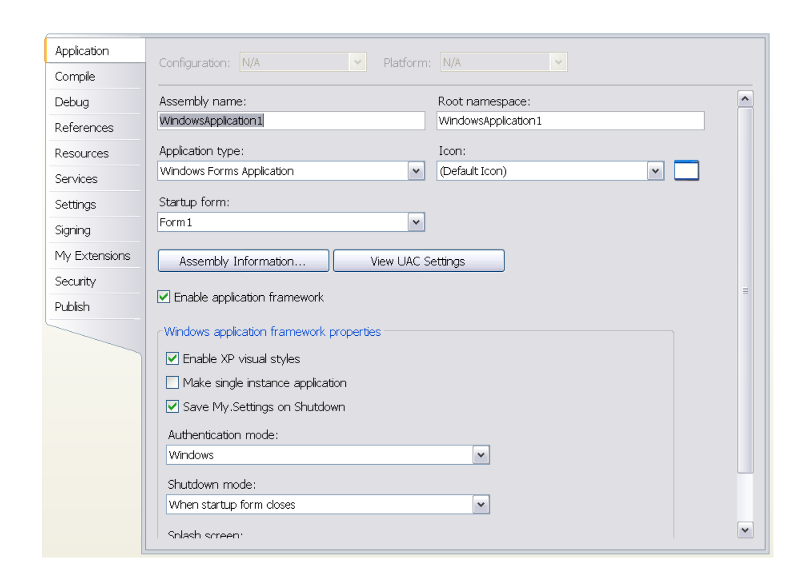
من خصائص عنصر التحكم PictureBox المفيدة، الخاصية AutoSize والتي تعني أن عنصر التحكم PictureBox من الممكن أن يتغير حجمه ليتوافق مع حجم الصورة. وكذلك الخاصية SizeMode والتي يمكنها أن تستخدم لضبط وضع الصورة في عنصر التحكم PictureBox فتسمح بتغيير أبعاد الصورة للتوافق مع عنصر التحكم.

قبل إضافة عنصر التحكم PictureBox، يجب إضافة الصورة إلى المشروع كمورد Resource. وبمجرد إضافة المورد إلى المشروع يمكنك استخدامه أكثر من مرة.

1.24.1.1إضافة الصورة كمورد Resource

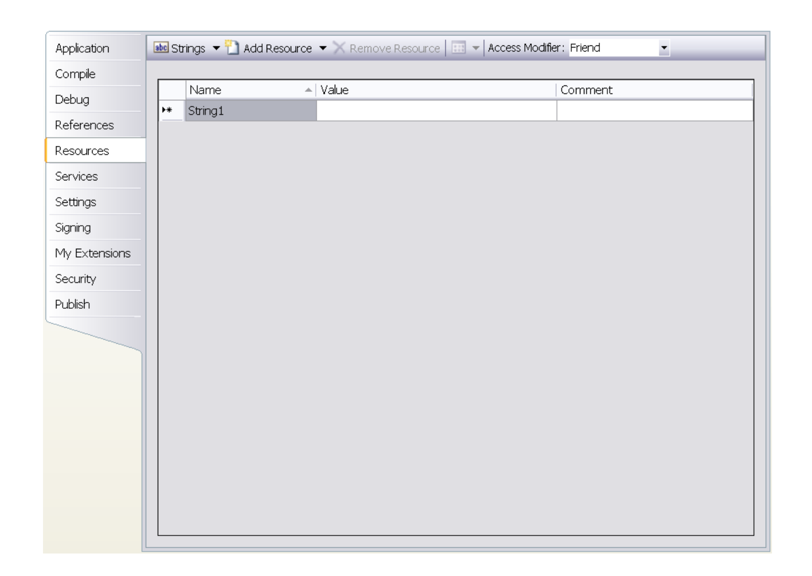
117. من قائمة File أختار New Project ومن تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

في نافذة Solution Explorer أنقر مرتين فوق أسم المشروع لتتفتح نافذة Project Designer – أنظر شكل 4-16.



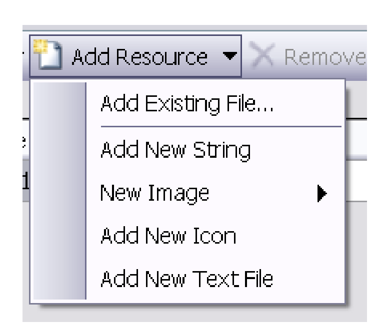
شكل 4-16: نافذة Project Designer.

أختار التبويب Resource من النافذة Project Designer – أنظر شكل 4-17.



شكل 4-17: التبويب Resource.

أنقر على السهم المجاور لـــ Add Resource وأختار منها Add Existing File – أنظر شكل 4-18.



شكل 4-18: إضافة مورد خارجي.

أستخدم النافذة التي سوف تظهر للاختيار صورة.

كرر الخطوة ٤ و ٥ مرة أخرى ليصبح عندك صورتين من ضمن موارد المشروع ظاهرتان في نافذة Solution Explorer.

1.24.1.2تدريب: عرض الصورة باستخدام عنصر التحكم *PictureBox*

118.أختار من Solution Explorer النموذج Form1 ثم أختار من قائمة View الاختيار Designer.

119.قم بسحب عنصر تحكم PictureBox من Toolbox ورميه فوق النموذج.

120.في نافذة الخصائص أنقر فوق المفتاح ... المجاور للخاصية Image لتظهر نافذة Select Resource.

121.أختر أحدى الصورتين.

122.أختار خاصية SizeMode وأضبطها لتصبح AutoSize.

123.أنقر مرتين فوق عنصر التحكم PictureBox لتنفتح نافذة Code Editor على عامل الحدث PictureBox1_Click.

124.بفرض أن الصورة الثانية التي أضفتها إلى الموارد أسمها MyPicName، قم بكتابة الشفرة التالية في طريقة الحدث.

```
PictureBox1.Image = My.Resources.MyPicName
```

125.أضرب المفتاح F5 لتنفيذ البرنامج، أنقر فوق الصورة وأنظر كيف تتغير الصورة.

1.24.2استخدام الصورة في خلفية النموذج

يمكن استخدام الصورة كخلفية للنموذج عوضاً عن استخدام الألوان. في التدريب التالي نتعلم معاً كيف نستخدم الصورة لتصبح هي خلفية النموذج.

1.24.2.1تدريب: استخدام الصورة في خلفية النموذج

126. من نافذة Solution Explorer أختار Form1.vb و من قائمة View أختار Designer.

127.أختار النموذج بالنقر خارج عنصر التحكم PictureBox.

128.في نافذة الخصائص، أنقر المفتاح ... للخاصية BackgroundImage حتي تنفتح نافذة Select Resource.

129.أختار أحدى الصورتين في الموارد ثم أنقر OK.

130.أختار خاصية BackgroundImageLayout ثم أضبط قيمتها لتصبح Stretch.

131.أضرب المفتاح F5 لتشاهد مظهر البرنامج الجديد.

1.25القوائم

سنتعلم الآن كيف ننشئ القوائم ونكتب الشفرات التي يتم تنفيذها عند اختيار القوائم. تمثل القوائم طريقة سهلة ومعتادة تسمح للمستخدمين بالوصول إلى الوظائف المختلفة برنامجك.

1.25.1إضافة القوائم

إن إضافة القوائم إلى البرامج باستخدام Visual Basic هو عمل بسيط. يمكنك استخدام عنصر التحكم MenuStrip الذي يظهر كمربع يحمل الكلمات Type Here ليظهر في الجانب الأيسر الأعلى من النموذج. حيث يمكنك أن تنقر هذا المربع وتكتب بداخله أسم القائمة.

عندما نقوم بضبط أسم قائمة، يمكن أن تظهر قوائم فرعية تحته وإلى يمين القائمة الأولى، لتسمح للمبرمج أن يقوم بتصميم قوائمه والقوائم الفرعية. عندما تنتهي تصميم قوائمك، يمكنك كتابة شفرة حدث أي من هذه القوائم.

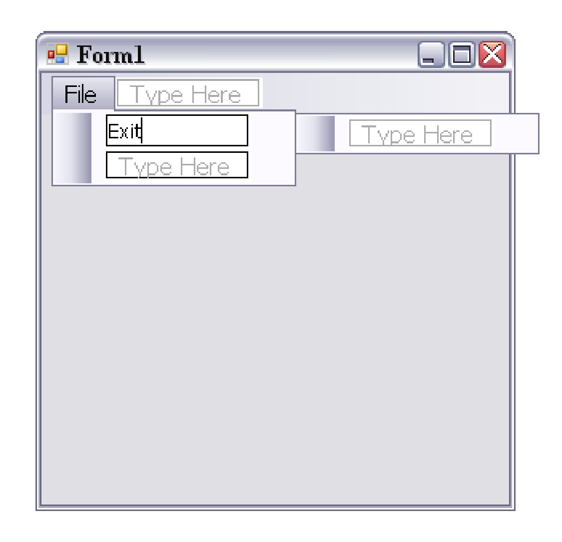
1.25.1.1تدريب: إضافة القائمة إلى نموذج

132. من قائمة File أختار New Project ومن تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

133.قم بسحب ورمي عنصر التحكم MenuStrip من Toolbox إلى النموذج، بغض النظر عن المكان الذي قمت برميه فيه سيحتل الحافة العلوية للنموذج وتظهر أول قائمة إلى اليسار.

134.أنقر فوق عنصر التحكم MenuStrip ثم قم بكتابة File ثم أضرب المفتاح Enter في لوحة المفاتيح.

135.في المربع الذي سوف يظهر تحت القائمة File أنقر فوقه ثم أكتب Exit ثم أضرب المفتاح Enter من لوحة المفاتيح – أنظر شكل 19-4.



شكل 4-19: تحرير القوائم.

136.أنقر مرتين فوق القائمة Exit لتظهر نافذة Code Editor.

137.في عامل التحكم ExitToolStripMenuItem_Clicl أكتب الشفرة التالية:

```
Application.Exit()
```

138.أضرب المفتاح F5 لتشغيل البرنامج ثم أختار Exit من القائمة File لتغلق البرنامج.

1.25.2استخدام خاصية Enabled

سنتعلم فيما يلي كيف نسمح للمستخدم أو لا نسمح له باستخدام القوائم أثناء التشغيل، حيث أنه في بعض الأحوال لا يمكن استخدام بعض الوظائف التي تقدمها القوائم. معظم البرامج تلجأ إلى عزل القائمة عوضاً عن إخفاءها حتى لا تسمح لمستخدميها باستخدام القائمة.

تستخدم الخاصية Enabled المرتبطة بعنصر التحكم StripMenu حتى تعزل أو لا تعزل استخدام القائمة. للخاصية Enabled قيمتين هما True وتعني أن القائمة غير معزولة والقيمة False وتعني أن القائمة معزولة.

1.25.2.1تدريب: استخدام الخاصية Enabled

استكمالا للتدريب السابق.

139.أضف للنموذج عنصر تحكم TextBox.

140.أضف إلى النموذج عنصر تحكم جديد MenuStrip ثم قم بتسميته Edit.

141.أنقر فوق المربع النازل من عنصر التحكم MenuStrip المسمى Edit ثم قم بتسميته Copy.

142. من نافذة الخصائص قم بضبط خاصية Enabled لعنصر التحكم CopyToolStripMenuItem لتصبح False.

143.أنقر فوق عنصر التحكم TextBox1 مرتين ثم أكتب الشفرة التالية:

```
If Textbox1.Text <> "" Then
 CopyToolStripMenuItem.Enabled = True
Else
 CopyToolStripMenuItem.Enabled = False
End If
```

144.قم بضرب المفتاح F5 في لوحة المفاتيح لتنفيذ البرنامج.

عند تنفيذ البرنامج مادام مربع النصوص فراغ من أي نص لن يمكنك استخدام Copy من القائمة Edit، فقط يمكنك استخدامها عندما يكون هذا المربع يحتوي على نص.

1.25.3إضافة شريط قوائم قياسي

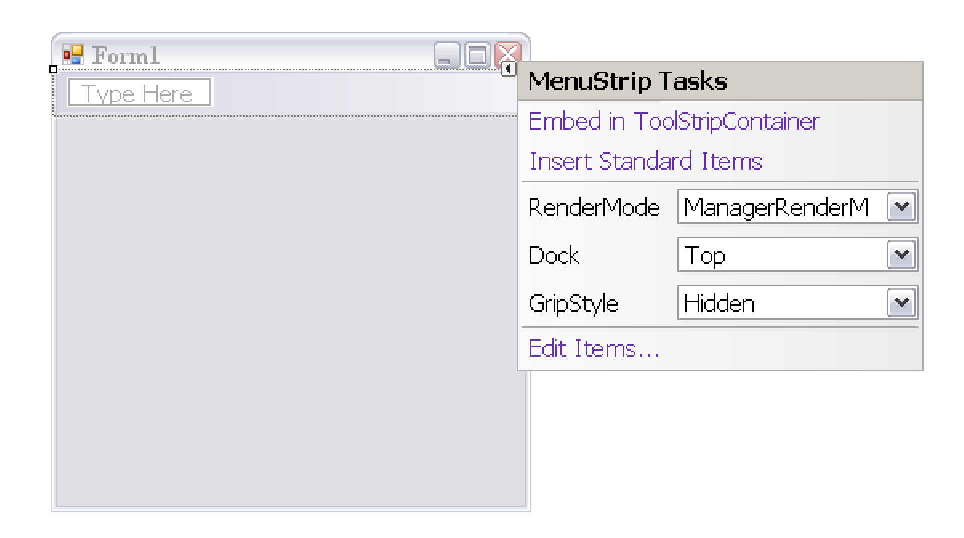
شريط القوائم القياسي هو شريط يحتوي القوائم القياسية في تطبيقات Microsoft Windows، وهي القوائم File و Edit و Tools و Help بما تضمنه من قوائم فرعية. يبسط Visual Basic.NET 2008 عملية تحرير القوائم القياسية على المبرمج حيث يمكن أضافتها بواسطة إجراءات بسيطة يبينها التدريب التالي.

1.25.3.1تدريب: إضافة شريط قوائم قياسي

145. من قائمة File أختار New Project ومن تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

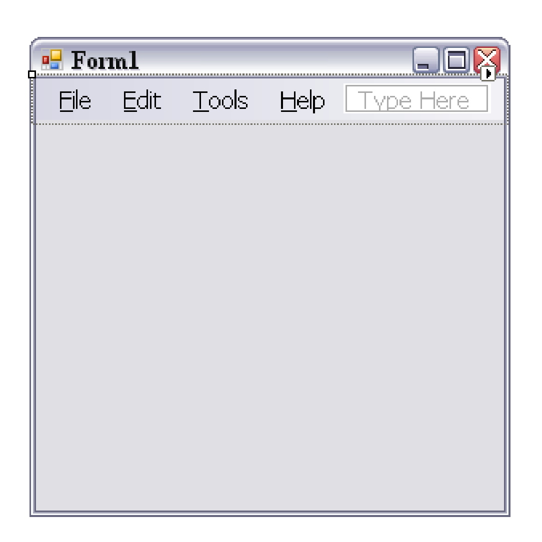
146.قم بسحب ورمي عنصر التحكم MenuStrip من Toolbox إلى النموذج، بغض النظر عن المكان الذي قمت برميه فيه سيحتل الحافة العلوية للنموذج وتظهر أول قائمة إلى اليسار.

147.أنقر فوق السهم الموجود في الجانب الأيمن العلوي من شريط القائمة لتظهر قائمة كالمبينة في شكل 4-20 اختار منها Insert Standard Items.



شكل 4-20: إضافة شريط قوائم قياسي إلى النموذج

148.يتحول شريط القوائم إلى شريط قوائم قياسي كالمبين في شكل 4-21.



شكل 4-21: النموذج مضاف إلية شريط القوائم القياسي

1.25.4القوائم المنسدلة Pop-Up Menus

كثير من البرامج تستخدم القوائم المنسدلة التي يمكن الوصول إليها عن طريق النقر فوق المفتاح الأيمن للفأرة. يمكن أنشاء القوائم المنسدلة في Visual Basic باستخدام عنصر التحكم ContextMenuStrip.

يتشابه عنصر التحكم ContextMenuStrip مع عنصر التحكم MenuStrip في طريقة عمله، إلا أنه يختلف عنه في أن عناصر القائمة لابد وأن توضع تحت عنصر القائمة الأول، أي أنها قوائم رئيسية.

أيضاً فإن عنصر التحكم ContextMenuStrip لابد أن يكون مرتبط مع نموذج أو عنصر تحكم آخر حيث ترغب في أن يظهر. يتم هذا عن طريق ضبط الخاصية ContextMenuStrip للنموذج أو عنصر التحكم الذي ترغب في أن تظهر القائمة المنسدلة فوقه. كما يمكنك أن تقوم بربط عنصر تحكم ContextMenuStrip في العديد من عناصر التحكم.

1.25.4.1 تدريب: عمل قائمة منسدلة وربطها بنموذج

149. من قائمة File أختار New Project ومن تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

150. قم بسحب ورمي عنصر التحكم ContextMenuStrip من Toolbox إلى سطح النموذج.

151. من نافذة خصائص النموذج قم بضبط الخاصية ContextMenuStrip لتصبح ContextMenuStrip1.

152. أنقر فوق عنصر التحكم ContextMenuStrip فوق النموذج ثم أكتب Option1 أضرب المفتاح Enter ثم بكتابة Option2 في المربع التالي ثم أضرب المفتاح Enter مرة أخرى.

153. أنقر مرتين فوق Option1 لتنفتح النافذة Code Editor.

154. في عامل الحدث Option1ToolStripMenuItem_Click أكتب الشفرة التالية:

```
MsgBox("You chose Option 1")
```

أختار من القائمة اليسرى لنافذة Code Editor عنصر التحكم Option2ToolStripMenuItem ومن لقائمة اليمنى الحدث Click.

في عامل الحدث Option2ToolStripMenuItem_Click أكتب الشفرة التالية:

```
MsgBox("You chose Option 2")
```

قم بضرب المفتاح F5 لتشغيل البرنامج.

1.26 استخدام عنصر التحكم Timer

في هذا القسم سنتناول كيفية استخدام عنصر التحكم Timer لتنفيذ إجراءات لا تستحثها مدخلات مستخدم التطبيق. مثل هذه الإجراءات التي لا تعتمد على مدخلات المستخدم ولكن تعتمد على مرور فترة معينة شائعة في العديد من التطبيقات مثل الحفظ الآلي Autosave للوثائق في برنامج Microsoft Word مثلا، أو تحديث صفحات الوب في بعض المتصفحات. مثل هذه الإجراءات يلزم لتنفيذها استخدام عنصر التحكم Timer.

يختلف عنصر التحكم Timer عن غيره من عناصر التحكم التي سبق وأن تناولنها حيث أنه ليس له واجهة استخدام. عناصر التحكم التي ليس لها واجهة استخدام يطلق عليها في Visual Basic.NET 2008 الاسم "**مكونات** *Components*".

المكون Timer له خاصيتين وحدث واحد دائماً ما يستخدموا.

الخاصية System.Timers.Timer.Enabled والتي تحدد ما إذا كان الـ Timer يعمل حيث تكون في هذه الحالة لها القيمة True أو لا تعمل وفي هذه الحالة تكون لها القيمة False.

الخاصية الثانية هي System.Timers.Timer.Interval والتي تحدد عدد المللي ثانية بين دقات Ticks ساعة Timer. على سبيل المثال إذا كانت هذه الخاصية مضبوطة على القيمة 1000، فإن الـ Timer سوف يقوم بدقة Tick كل ١٠٠٠ مللي ثانية أي كل ثانية.

حدث الدقة أو Tick Event هو الحدث الذي سوف يحدث آلياً عند مرور الـ Interval المحدد من قبل المبرمج، ويمكن بضبط كلاً من خاصية Enabled وخاصية Interval وإضافة تعليمات للحدث Tick تكرار إجراءات الحدث Tick على فترات زمنية متساوية وتساوي الفترة المحددة في Interval.

1.26.1.1 تدريب: استخدام المكون *Timer* في بناء تطبيق الساعة

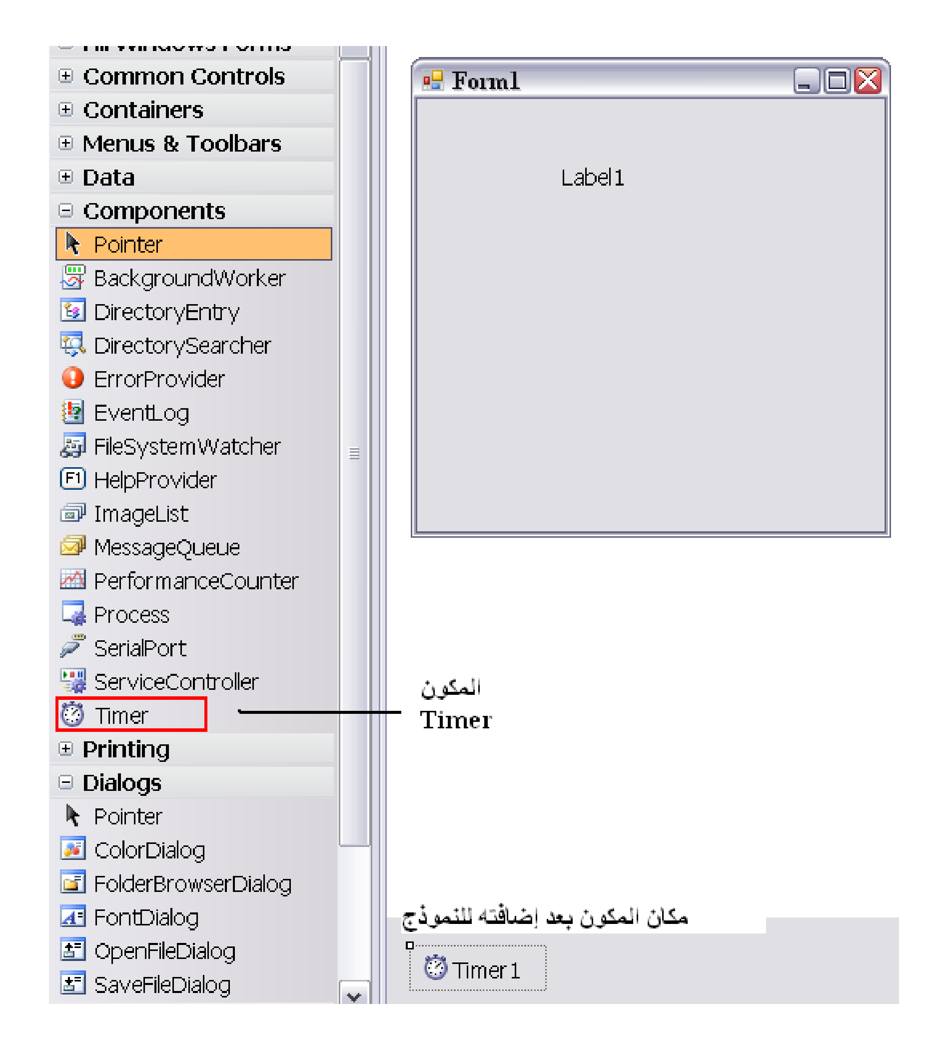
155. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

156. قم بسحب عنصر تحكم من النوع Label ورميه فوق النموذج Form1 سيحمل الاسم Label1 آلياً.

157. من Toolbox أنقر التبويب Components ثم أسحب من تحته المكون Timer و قد برميه فوق النموذج Form1 يصبح اسمه آلياً Timer1 كما أنه لا يظهر فوق الـ Form1 لكنه يظهر في شريط أسفل Form1 كما هو مبين في شكل 22-4 حيث أنه يفتقر إلى واجهة الاستخدام فلا محل له فوق النموذج.

158. أنتقل إلى نافذة الخصائص وتأكد أن عنصر التحكم موضوع النافذة هو Timer1 قم بضبط الخاصية Enabled لتصبح True والخاصية Interval لتصبح ١٠٠٠.

159. أنقر المكون Timer1 نقراً مزدوجاً لتتمكن من فتح نافذة Code Editor على الحدث Timer1_Tick.



شكل 4-22: المكون Timer في Toolbox وعلى النموذج Form1

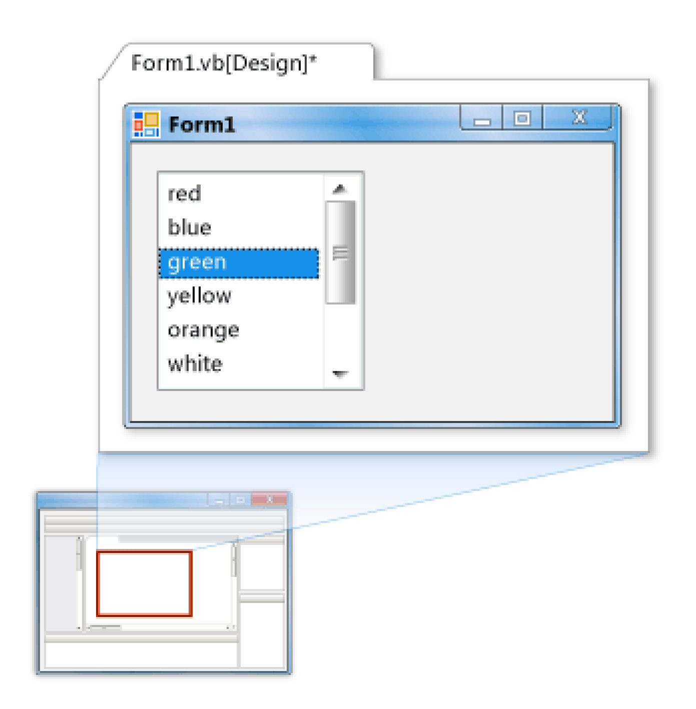
أضف شفرة التعليمات التالية:

```
Label1.Text=My.Computer.Clock.LocalTime.ToLongTimeString
```

اضرب المفتاح F5 لتنفيذ البرنامج.

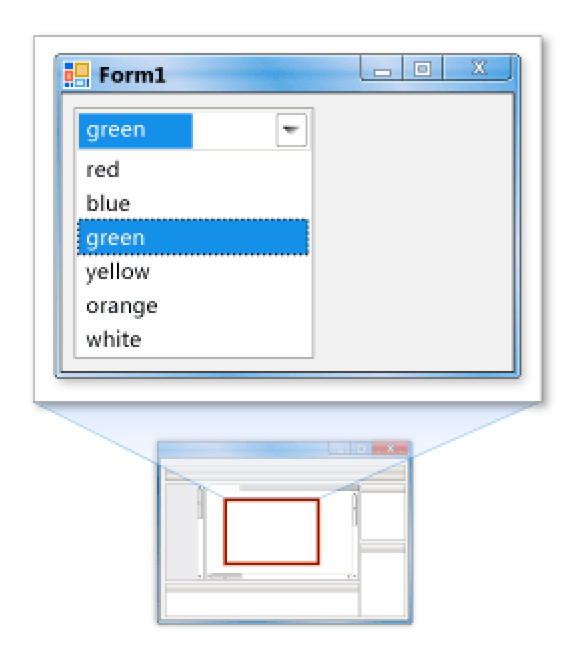
1.27عناصر التحكم ListBoc و ComboBox

عنصري التحكم ListBox و ComboBox يسمحا للمستخدم أن يختار عنصر item من قائمة من العناصر. يختلف عنصر التحكم ListBox عن عنصر التحكم ComboBox بأنه يعرض عدة أسطر في آن واحد، على حين لا يعرض عنصر التحكم ComboBox إلا عنصر واحد فقط.



شكل 4-23: عنصر التحكم ListBox يعرض أكثر من بند في آن واحد

بينما يتميز عنصر التحكم ComboBox بأنه يسمح للمستخدم أن يختار من قائمة العناصر أو يضيف إليها ويختار ما أضافه مما يجعله مزيج من الـ TextBox والـ ListBox في نفس الوقت.



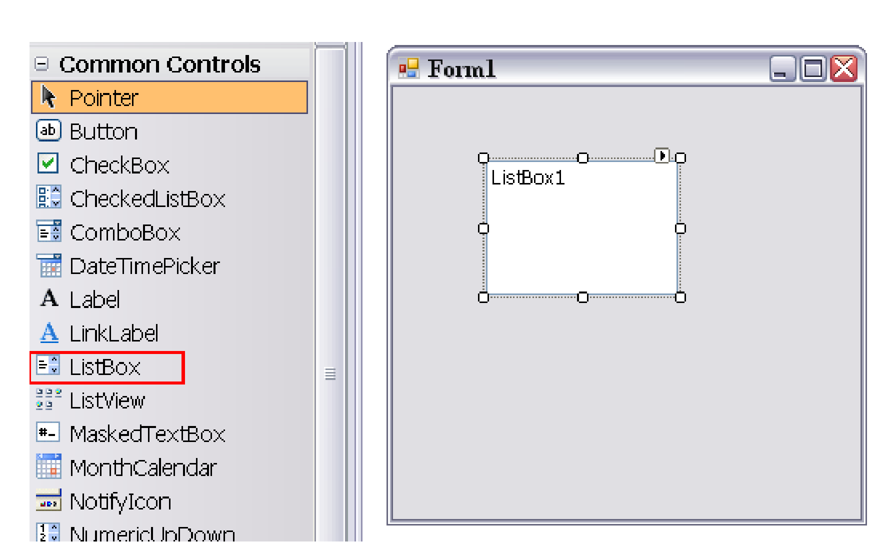
شكل 4-24: عنصر التحكم ComboBox يسمح للمستخدم أن يختار أو أن يحرر البند الذي يختاره

في هذا القسم سنتعلم كيف نضيف أو نحذف عناصر items إلى عناصر تحكم القوائم بنوعيها، كما سنتعلم كيف نبني حدث يستجيب به البرنامج لاختيار بند معين من القوائم.

1.27.1تدريب: إضافة عنصر Item إلى عنصر التحكم ListBox

160.من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

161.قم بسحب عنصر تحكم من النوع ListBox ورميه فوق النموذج Form1 سيحمل الاسم ListBaox1 آلياً.



شكل 4-25: إضافة عنصر التحكم ListBox

162.أنقر مرتين فوق النموذج لتفتح نافذة تحرير التعليمات. معالج الحدث الافتراضي الذي سوف يظهر هو Form1_Load أي حدث تحميل النموذج في ذاكرة البرنامج.

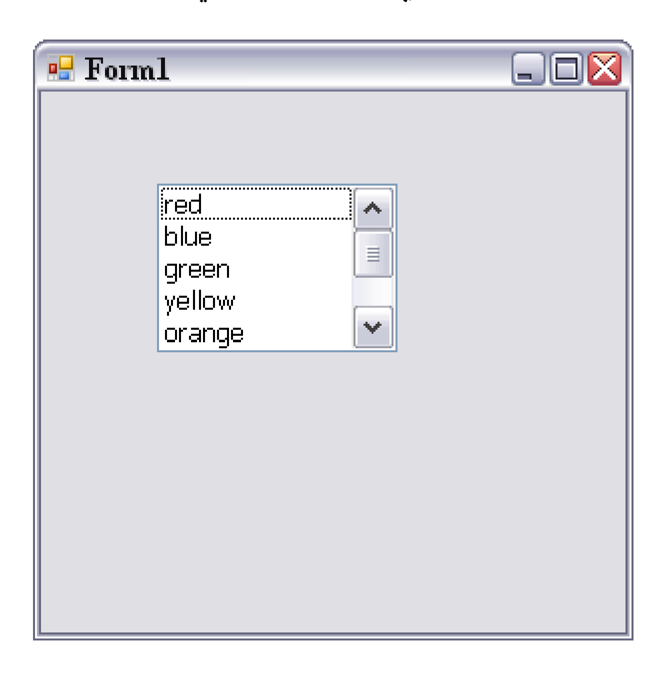
163.أكتب التعليمات التالية في هذا الحدث.

```
With Me.ListBox1.Items
   .Add("red”)
   .Add("blue”)
   .Add("green”)
   .Add("yellow”)
   .Add("orange”)
```

```
    .Add("pink")
    .Add("brown")
    .Add("black")
End With
```

أنقر F5 لتنفيذ البرنامج.

لاحظ أن القائمة أصبحت تحتوي على الألوان التي قمنا بكتابتها.



شكل 4-26: التطبيق أثناء التشغيل

*1.27.1.1*ملاحظات حول تعليمات البرنامج

لعل القارئ لاحظ أن هناك تعليمات جديدة استخدمت في هذا التطبيق، لذلك سنتوقف عند هذه التعليمات قليلا.

القسم التالي من التعليمات تحديداً هو ما يجب أن نتوقف عنده.

```
With Me.ListBox1.Items
    ......... .
End With
```

فالكتلة With .... End With تستخدم عندما سوف نقوم بمجموعة من العلميات على كائن ما. وهذا الكائن في هذه الحالة هو Me.ListBox1. أي أنه ListBox1 لكن لماذا قمنا بكتابته بهذه الطريقة؟ Me تعني النموذج نفسه، فكأننا نقول عنصر التحكم ListBox1 الموجود في النموذج Form1.

إن هذه التعليمات يمكن كتابتها بصورة أخرى هي كالتالي

```
Me.ListBox1.Add("red")
Me.ListBox1.Add("green")
Me.ListBox1.Add("blue")
Me.ListBox1.Add("yellow")
Me.ListBox1.Add("orange")
Me.ListBox1.Add("pink")
Me.ListBox1.Add("brown")
Me.ListBox1.Add("black")
```

وهكذا نكون استغنينا عن With ...End With لكننا قمنا بالكثير من كتابة التعليمات.

1.27.1.2 الطريقة *Add*

الطريقة Add تستخدم مع كلاً من عنصر التحكم ListBox وعنصر التحكم ComboBox لأضافه البنود، حيث تكتب بهذه الطريقة

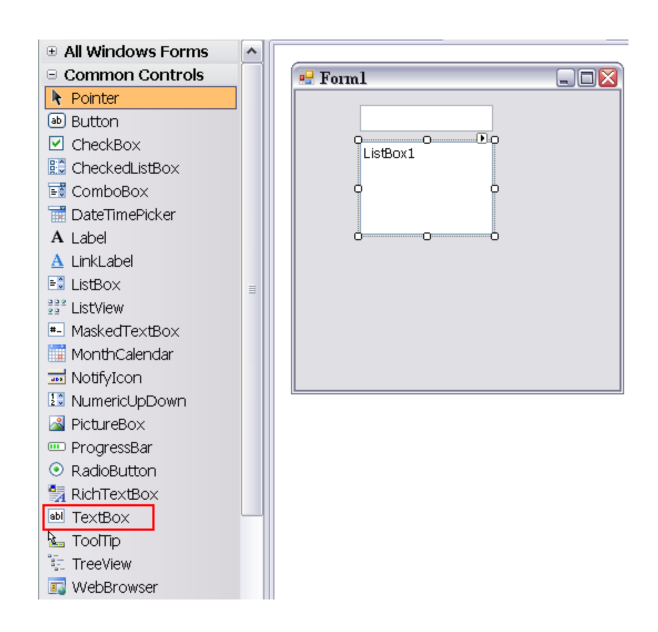
```
Control.Add("item")
```

ومن الواضح أن العناصر لابد أن تكون قيم نصية.

1.27.2 تدريب: تصميم حدث للاستجابة على اختيار بند من القائمة

استكمالاً على التطبيق السابق.

164. قم بسحب عنصر تحكم من النوع Textbox ورميه فوق النموذج Form1 سيحمل الاسم Textbox1 آلياً.



شكل 4-27: إضافة عنصر التحكم TextBox إلى واجهة التطبيق

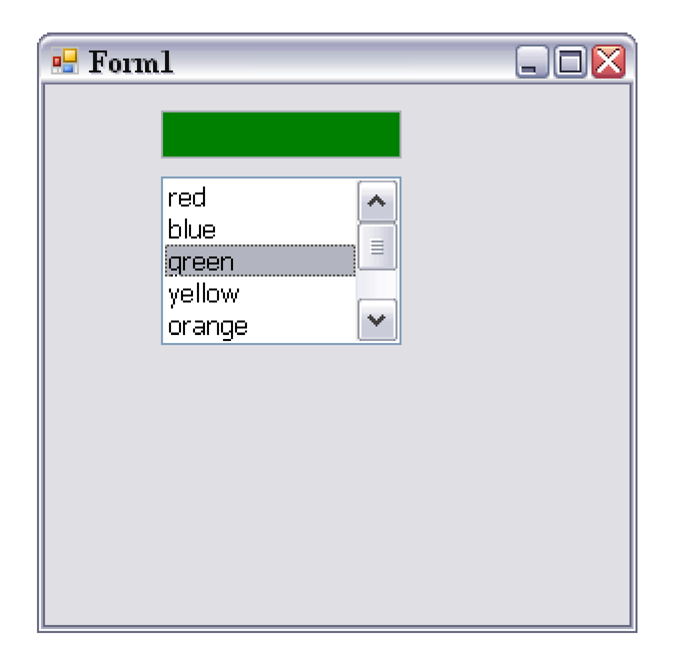
أنقر مرتين فوق ListBox1 لتفتح نافذة تحرير التعليمات. معالج الحدث الافتراضي الذي سوف يظهر هو ListBox1_SelectedIndexChanged أي حدث اختيار عنصر من القائمة.

أكتب التعليمات التالية في الحدث.

```
Dim ColorName As String = CStr(ListBox1.SelectedItem)

If ColorName IsNot Nothing Then
  Me.TextBox1.BackColor = _
    System.Drawing.Color.FromName(ColorName(
End If
```

أنقر على F5 لتنفيذ البرنامج. قم باختيار أحد الألوان ليتغير لون خلفية الـ Textbox تبعاً للون الذي قمت باختياره.



شكل 4-28: آثر اختيار اللون الأخضر من ListBox1 على خلفية Textbox1

1.27.2.1 ملاحظات حول تعليمات البرنامج

في هذا البرنامج هناك بعض التعليمات الجديدة المستخدمة، وفيما يلي سنتعرف على هذه التعليمات وما الذي تقوم به.

في السطر الأول:

```
Dim ColorName As String = CStr(ListBox1.SelectedItem)
```

قمنا بالإعلان عن متغير ColorName من النوع الـ String، ثم استخدمنا دالة التحويل Casting إلى لفظي CStr لتحويل العنصر المختار SelectedItem من قائمة عناصر عنصر التحكم ListBox1.

ثم استخدمنا عبارة if محلوقة بشرط يتكون من ColorName وهو أسم المتغير الذي أعلنا عنه من قبل واختزنا فيه قيمة نص العنصر المختار من ListBox1. ثم IsNot وهي دالة تستخدم للمقارنة بين قيمتين، القيمة الأولى هي ColorName والثانية هي Nothing وهي دالة أخرى تمثل الـ Null

```
If ColorName IsNot Nothing Then
      ......
End If
```

وهذا يعني أنه إذا كانت قيمة ColorName ليست Null فإن الشرط يتحقق، لأن IsNot ترجع قيمة Boolean هي True عند عدم التساوي و False عند التساوي.

وعند تحقق الشرط يتم تنفيذ التعليمات التالية:

```
Me.TextBox1.BackColor = _
   System.Drawing.Color.FromName(ColorName(
```

تعرفنا فيما سبق على Me والتي تعني الـ Form1 نفسه، ثم TextBox1 وهو أسم العنصر المطلوب تغيير خاصيته BackColor ثم علامة التخصيص.

الشرطة السفلية _ تعني أن السطر التالي هو مكمل للسطر الذي انتهى بهذه العلامة.

ثم تخصيص لون باعتماد مكتبة ألوان خاصة سنتعرف عليها في موضع متقدم من هذا الكتاب.

1.27.3تدريب: تعيين ما إذا كان عنصر ما موجود في القائمة أو لا

عند ما نقوم بإضافة عنصر إلى قائمة، فإنه غالباً ما لا نريد أن يكون هذا العنصر مضاف من قبل. في التدريب التالي نقوم بنسخ عنصر من ListBiox إلى عنصر تحكم سوف نضيفه على مشروعنا هو ComboBox بأن ننقر عليه مرتين في ListBox.

أضف عنصر تحكم ComboBox بأن تسحبه من الـ Toolbox ثم ترميه فوق النموذج. يصبح الاسم الافتراضي لهذا النموذج هو ComboBox1.

أنتقل إلى محرر التعليمات Code Editor.

في محرر التعليمات أختار من قائ مة Class Name عنصـ ر التحـ كم ListBox1.

من قائمة Method Name أختار Double Click.



شكل 4-29: أختار ListBox من Class Name وأختر Double Click من Method Name.

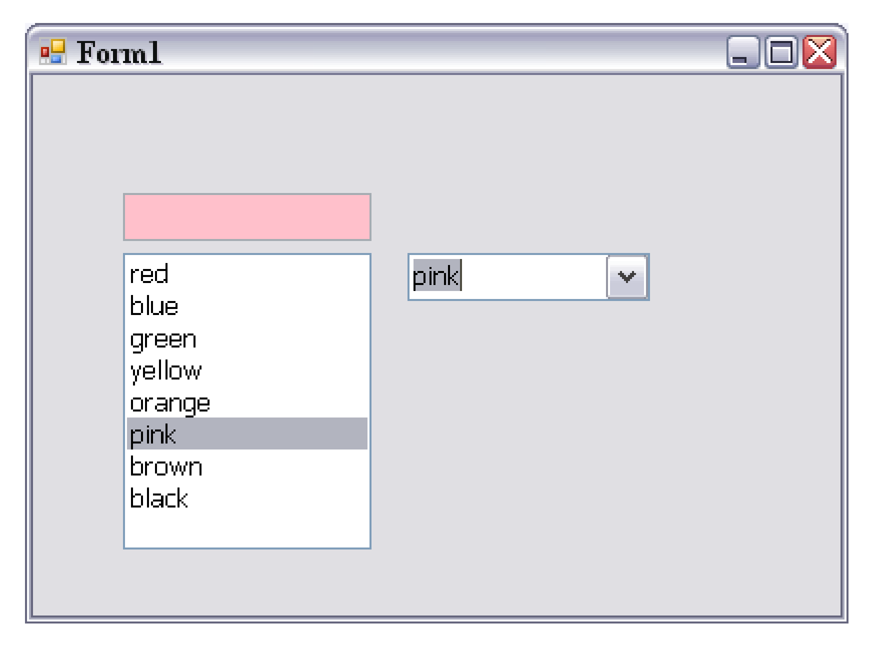
في الإجراء الذي سوف يظهر بعنوان ListBox1_DoubleClick قم بكتابة التعليمات التالية:

```
If ListBox1.SelectedItem IsNot Nothing Then
  If Not _
 ComboBox1.Items.Contains(Me.ListBox1.SelectedItem)_
Then
    Me.ComboBox1.Items.Add(Me.ListBox1.SelectedItem(
  End If

  Me.ComboBox1.SelectedItem = Me.ListBox1.SelectedItem

End If
```

قم بضرب المفتاح F5 لتنفيذ البرنامج.



شكل 4-30: البرنامج بعد تعديله أثناء التشغيل.

*1.27.3.1*تعليق على تعليمات البرنامج

ما سوف نعلق عليه في هذا الجزء هو السطر

```
ComboBox1.Items.Contains(Me.ListBox1.SelectedItem)
```

و هو يعني أن الـ ComboBox1 يضم Contains ضمن عناصره Items عنصر باسم Me.ListBox1.SelectedItem.

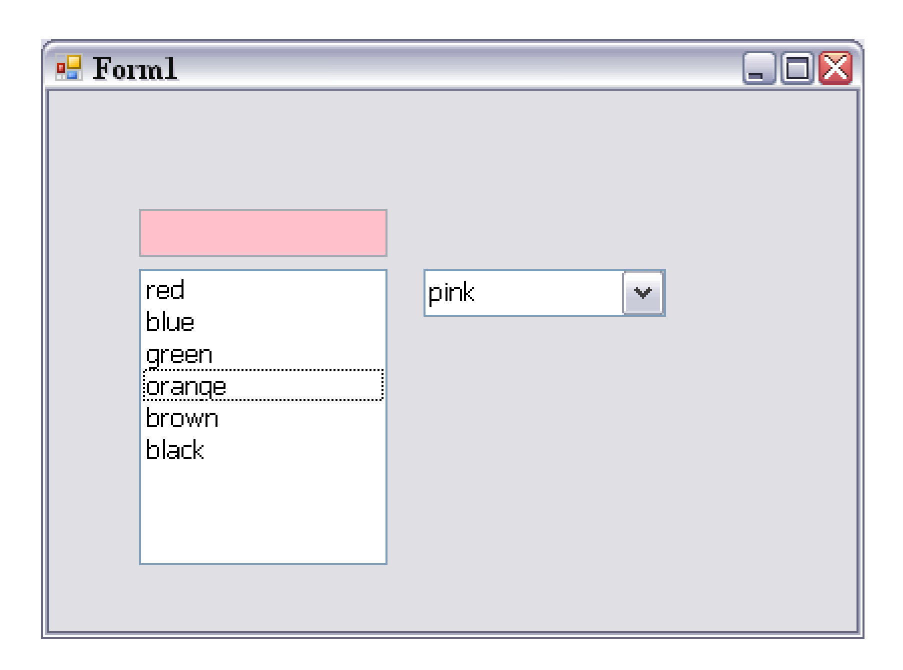
1.27.4حذف عنصر من قائمة

في التدريب التالي سوف نتعلم حذف عنصر من قائمة باستخدام الطريقة Remove.

أضف التعليمات التالية فوق End If في حدث ListBox1_DoubleClick.

```
Me.ListBox1.Items.Remove(Me.ListBox1.SelectedItem)
```

أضرب المفتاح F5 لتنفيذ البرنامج.



شكل 4-31: حذف العنصر Pink من قائمة ListBox، إضافته إلى ComboBox.

1.28استخدام عناصر التحكم MonthCalendar و DateTimePicker

في هذا القسم سوف نتعلم كيف نعرض التاريخ فوق النموذج، وكيف يمكننا أن ننشأ حدث يعتمد على اختيار مستخدم التطبيق للتاريخ.

ع ند ا ستخدام عنا صر التح كم ال تي تظ هر التار يخ فإن هذا يضد من لل مبرمج أن مستخدمي تطبيقه سوف يدخلون التاريخ بالهيئة المقبولة في Visual basic.Net.

تحتوي Visual Basic.Net 2008 على عنصر ي تح كم للتار يخ ه ما MonthCalender و DateTimePicker.

عنصر التحكم MonthCalender: حيث يظهر روزنامة لشهر محدد يمكن للمستخدم أن يختار تاريخ محدد أو مدى من التواريخ.

عنصر التحكم DateTimePicker: ويظهر في حالتين، الحالة الافتراضية هو أن يظهر في صورة صندوق نصوص وإلى جواره سهم قائمة منسدلة، عند النقر عليه

تظهر الروزنامة و من ثم يتم اختيار التاريخ المطلوب. والحالة الثانية هو أن يعرض عنصر التحكم الزمن بدل التاريخ.

1.28.1تدريب: استرجاع البيانات من MonthCalender وعرضها في Label

165.من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

166.قم بسحب عنصر تحكم من النوع Label ورميه فوق النموذج Form1 سيحمل الاسم Label1 آلياً.

167.من نافذة Properties أجعل خاصية Text لعنصر التحكم Label1 فارغة.

168.قم بسحب عنصر تحكم من النوع MonthCalender ورميه فوق النموذج Form1 سيحمل الاسم MonthCalender1 آلياً.

أنقر مرتين فوق عنصر التحكم MonthCalender1 لتقوم بفتح محرر التعليمات على الحدث الافتراضي لهذا العنصر و هو الحدث MonthCalendar1_DateChanged.

169.أضف التعليمات التالية لهذا الحدث.

```
Me.Label1.Text = _
 CStr(Me.MonthCalendar1.SelectionRange.Start(
```

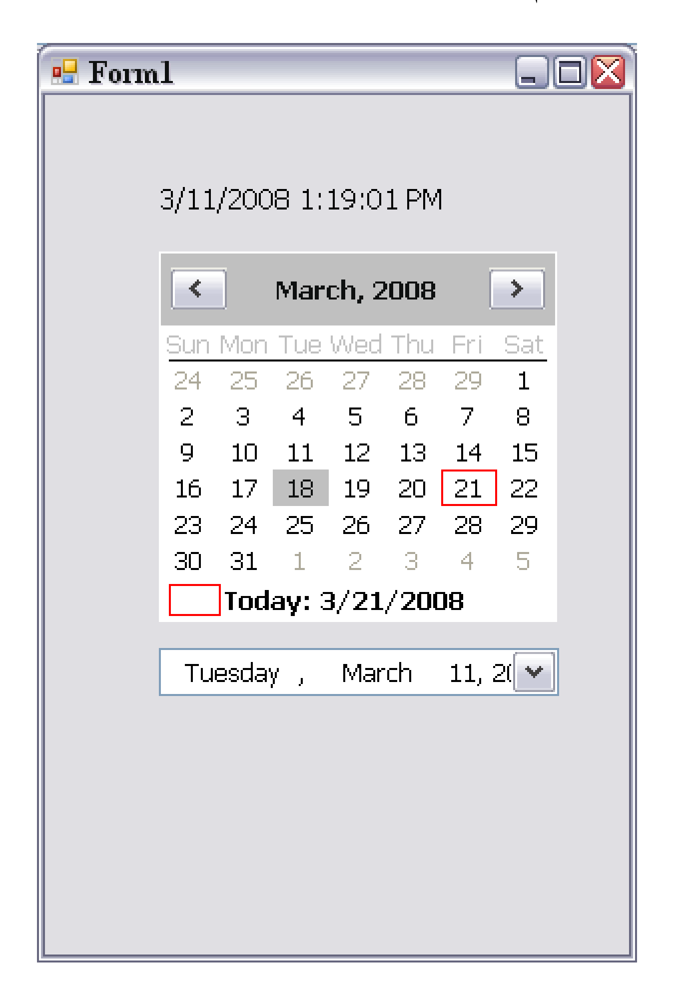
170. قم بسحب عنصر تحكم من النوع DateTimePicker ورميه فوق النموذج Form1 سيحمل الاسم DateTimePicker1 آلياً.

أنقر مرتين فوق عنصر التحكم DateTimePicker1 لتقوم بفتح محرر التعليمات على الحدث الافتراضي لهذا العنصر و هو الحدث DateTimePicker1_ValueChanged.

171.أضف التعليمات التالية لهذا الحدث.

```
Me.Label1.Text = _
     CStr(Me.DateTimePicker1.Value)
```

قم بضرب المفتاح F5 لتنفيذ البرنامج، ثم جرب اختيار تاريخ من الـ MonthCalender1 ثم من DateTimePicker1.



شكل 4-32: واجهة التطبيق.

1.28.2تدريب: استرجاع عدة بيانات تاريخ

في التدريب السابق استخدمنا SelectedRange.Start لتكون القيمة المرتجعة من اختيار التاريخ في عنصر التحكم MonthCalender1 هو أول تاريخ في يختاره المستخدم، لكننا اشرنا فيما سبق أنه يمكن للمستخدم أن يختار نطاق Range من التاريخ، و في هذه الحالة يلزمه أن يختار الخاصيتين SelectedRange.Start و SelectedRange.End.

العدد الافتراضي المتاح اختياره من الأيام هو سبعة أيام لكن يمكن تغيير هذه القيمة من خلال الخاصية MaxSelectionCount.

قم بتغيير التعليمات الموجودة في الحدث MonthCalendar1_DateChanged ليصبح كالتالي:

```
Me.MonthCalendar1.MaxSelectionCount = 14

If Me.MonthCalendar1.SelectionRange.Start = _
   Me.MonthCalendar1.SelectionRange.End Then

   Me.Label1.Text = _
     CStr(Me.MonthCalendar1.SelectionStart)

Else

   Me.Label1.Text = _
     Me.MonthCalendar1.SelectionRange.Start & _
    " - " & Me.MonthCalendar1.SelectionRange.End

End If
```

هذه التعليمات تجعل القيمة القصوى الممكن اختيار ها من الأيام هي أربعة عشرة يوم. ثم تستخدم الجملة If لاختبار ما إذا كان النطاق المختار في MonthCalender1 للخاصيتين SelectedRange.Start و SelectedRange.End لهما نفس القيمة (أي أن المستخدم أختار يوم واحد) أم لا (المستخدم أختار نطاق من الأيام)، وفي حالة تحقق الشرط، فإن Label1 يعرض تاريخ يوم واحد فقط، أما في حالة عدم تحقق الشرط فسوف يعرض Label1 تاريخ البداية والنهاية.

أضرب المفتاح F5، اختبر تنفيذ البرنامج.

1.28.3شكل بيانات التاريخ

في كثير من الأحيان قد يحتاج المبرمج إلى تغيير الشكل الذي تظهر به بيانات التاريخ، و في هذه الحالة فإنه يحتاج أن يقوم با ستخدام الوظيفة FormateDateTime وإحدى القيم المبينة في الجدول اللاحق:

القيمة	مثال
DateFormat.GeneralDate	11/22/1963 12:00
DateFormat.LongDate	Friday, November 22, 1963
DateFormat.ShortDate	11/22/1963
DateFormat.LongTime	12:00:00 PM

DateFormat.ShortTime	12:00

1.28.4تدريب: تعديل شكل بيانات التاريخ

قم بتغيير التعليمات الموجودة في الحدث MonthCalendar1_DateChanged ليصبح كالتالي:

```
Me.MonthCalendar1.MaxSelectionCount = 14

If Me.MonthCalendar1.SelectionRange.Start = _
   Me.MonthCalendar1.SelectionRange.End Then

   Me.Label1.Text = FormatDateTime( _
      Me.MonthCalendar1.SelectionStart, _
      DateFormat.LongDate)
Else
   Me.Label1.Text = FormatDateTime( _
      Me.MonthCalendar1.SelectionRange.Start, _
      DateFormat.LongDate) & " - " & FormatDateTime( _
      Me.MonthCalendar1.SelectionRange.End,
DateFormat.LongDate)
End If
```

كما يلاحظ القارئ، فإن القيمة التي سوف تظهر Label1 هي القيمة المختارة من MonthCalender1 لكن بعد تمرير ها من خلال الوظيفة FormatDateTime وتعيين شكل البيانات المرتج ليكون DateFormat.LongDate.

الصورة العامة للوظيفة FormateDateTime هي كالتالي:

(FormatDateTime(value, constant حيث value هي قيمة التاريخ المراد إعادة صياغة شكله، بينما constant هو أحد القيم من الجدول السابق.

أضرب المفتاح F5 لتنفيذ البرنامج.

1.29 المكون ErrorProvider

عاملنا مع المكونات Components سابقاً حينما قدمنا المكون Timer. في هذا القسم نقدم مكون آخر هو ErrorProvider. الوظيفة الأساسية للمكون ErrorProvider مساعدة المبرمج ومستخدم التطبيق على التأكد أن مدخلات البرنامج مطابقة لتصميمه.

1.29.1 تدريب: التحقق من المدخلات باستخدام ErrorProvider

في هذا التدريب سنستخدم المكون ErrorProvider للتحقق من مطابقة مدخلات التطبيق عبر مجموعة من صناديق النصوص TextBox للغرض التصميمي.

172. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

173. قم بسحب عنصر تحكم من النوع Label ورميه فوق النموذج Form1 سيحمل الاسم Label1 آلياً.

174. قم بتغيير قيمة الخاصية Text لعنصر التحكم Label1 لتصبح Name.

175. قم بسحب عنصر تحكم من النوع Label ورميه فوق النموذج Form1 سيحمل الاسم Label2 آلياً.

176. قم بتغيير قيمة الخاصية Text لعنصر التحكم Label2 لتصبح Age.

177. أضف TextBox أمام كل عنصر تحكم Label ليصبح لديك TextBox1 و TextBox2.

178. قم بسحب مكون من ErrorProvider من تبويب Components في الـ Toolbox ورميه فوق النموذج Form1 سيحمل الاسم ErrorProvider1 آلياً.

179. افتح محرر التعليمات.

180. من قائمة Class name أختار TextBox2.

181. من قائمة Method Name أختار Validating.

أضف التعليمات التالية إلى الحدث TextBox2_Validating:

```
If Not IsNumeric(TextBox2.Text) Then
   ErrorProvider1.SetError(TextBox2, _
      "You must enter a numeric value.")
Else
   ErrorProvider1.SetError(TextBox2, "")
End If
```

اضرب المفتاح F5 لتنفيذ البرنامج واختباره.

1.29.2 تعليق على تعليمات التدريب

استخدمنا في هي التعليمات الوظيفة IsNumeric والتي ترجع قيمة true إذا كان ما بين القوسين رقمي وقيمة false إذا كان ما بين القوسين غير رقمي.

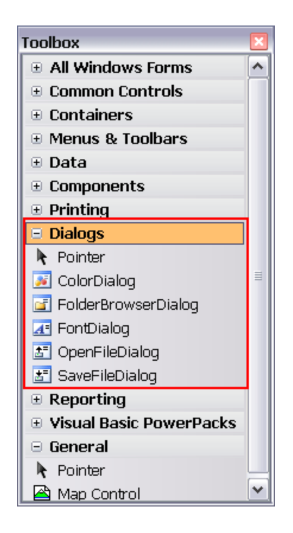
في حال عدم تحقق شرط أن قيمة TextBox2 هي قيمة رقمية، فإن الكائن ErrorProvider سوف يظهر علامة تنبيه إلى جوار الكائن TextBox2 عند الإشارة إلى هذه العلامة بمؤشر الفارة تظهر الرسالة You must enter a numeric value. وذلك نتيجة استخدام التعليمات في الصورة التالية:

```
ErrorProvider1.SetError(TextBox2, _
      "You must enter a numeric value.")
```

حيث يتم أضافه بين الأقواس بعد SetError اسم الكائن الذي سوف تظهر إلى جواره علامة التنبيه، ثم الرسالة التي سوف تظهر عند الإشارة إلى علامة التنبيه.

1.30 استخدام صناديق الحوار Dialog Boxes

يتوفر في Visual Basic.Net 2008 مجموعة من عناصر التحكم التي تيسر على المستخدم عدد من العمليات الروتينية في التطبيقات العاملة في بيئة التشغيل Microsoft Windows مثل نافذة فتح الملفات أو نافذة اختيار الألوان. ويطلق عل هذه العائلة من أدوات التحكم اسم صناديق الحوار Dialog Boxes، وهي التي يمكن الاطلاع عليها في التبويب Dialogs في الـ Toolbox كما هو مبين في شكل 33-4.



شكل 4-33: تبويب Dialogs

ويجب أن ننوه أن هذه العائلة من عناصر التحكم هي من جنس المكونات أي أنها لا تظهر فوق النموذج.

1.30.1تدريب: استخدام صندوق الحوار FolderBrowserDialog

182. من قائمة File أختار New Project و من تبو يب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

183. قم بسحب عنصر تحكم من النوع Label ورميه فوق النموذج Form1 سيحمل الاسم Label1 آلياً.

184. قم بسحب عنصر تحكم من النوع Button ورميه فوق النموذج Form1 ثم قم بتغيير خصائصه للتوافق مع الجدول التالي.

الخاصية	القيمة
Name	FolderPath
Text	Path

185. قم بسحب عنصر تحكم من النوع FolderBrowserDialog ورميه فوق النموذج Form1 سيحمل الاسم FolderBrowserDialog1 آلياً.

أ   قر مرة   ين فوق المفتاح لتقوم بف   تح محرر التعليمات على الحدث FolderPath_Click.

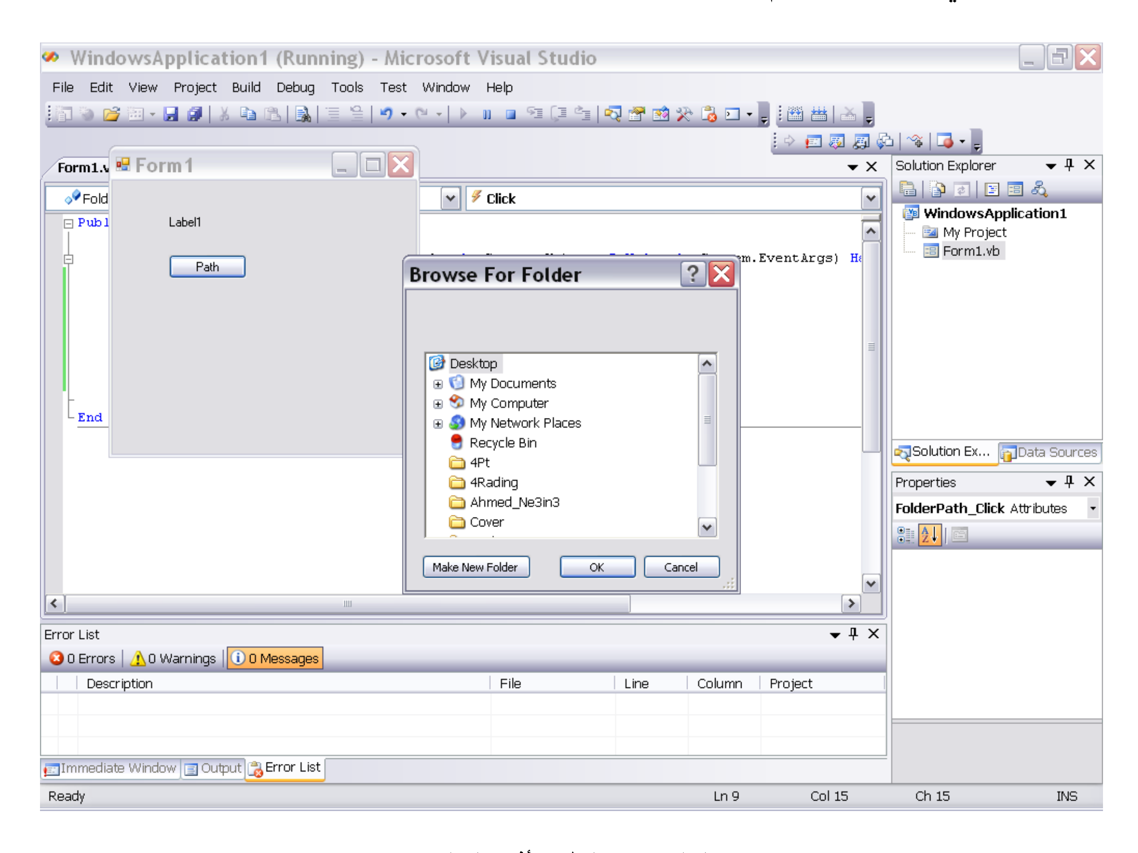
أضف التعليمات التالية للحدث FolderPath_Click:

```
If FolderBrowserDialog1.ShowDialog() = _
   Windows.Forms.DialogResult.OK Then

   Label1.Text = FolderBrowserDialog1.SelectedPath

End If
```

أضرب المفتاح F5 من لوحة المفاتيح لتنفيذ البرنامج. عند تنفيذ البرنامج أنقر المفتاح Path لتظهر نافذة تصفح المجلدات أختار مجلداً و أنقر OK لينطبع عنوان المجلد في عنصر التحكم Laberl1.



شكل 4-34: التطبيق أثناء العمل.

لتفسير التعليمات المستخدمة في هذا التدريب

في البداية استخدمنا العبارة IF لاختبار الشرط:

```
FolderBrowserDialog1.ShowDialog() = _
   Windows.Forms.DialogResult.OK Then
```

و هو شرط نوعاً ما مركب حيث يقوم بالتالي:

أظهار واجهة عنصر التحكم FolderBrowserDialog1 و هي التعليمات

```
FolderBrowserDialog1.ShowDialog()
```

اختبار ما إذا كان مرجوع عنصر التحكم هو

```
Windows.Forms.DialogResult.OK
```

وهو الذي يعني النقر فوق المفتاح OK.

وعند تحقق هذا الشرط، فإن الخاصية Text لعنصر التحكم Label1 تصبح تساوي:

```
FolderBrowserDialog1.SelectedPath
```

وهو ما يعني أن هذه الخاصية سوف يختزن فيها المجلد المختار من عنصر التحكم FolderBrowserDialog1

1.30.2 تدريب: استخدام صندوق الحوار FontDialog

186. قم بسحب عنصر تحكم من النوع Button ورميه فوق النموذج Form1 ثم قم بتغيير خصائصه للتوافق مع الجدول التالي.

الخاصية	القيمة
Name	TextFont
Text	Font

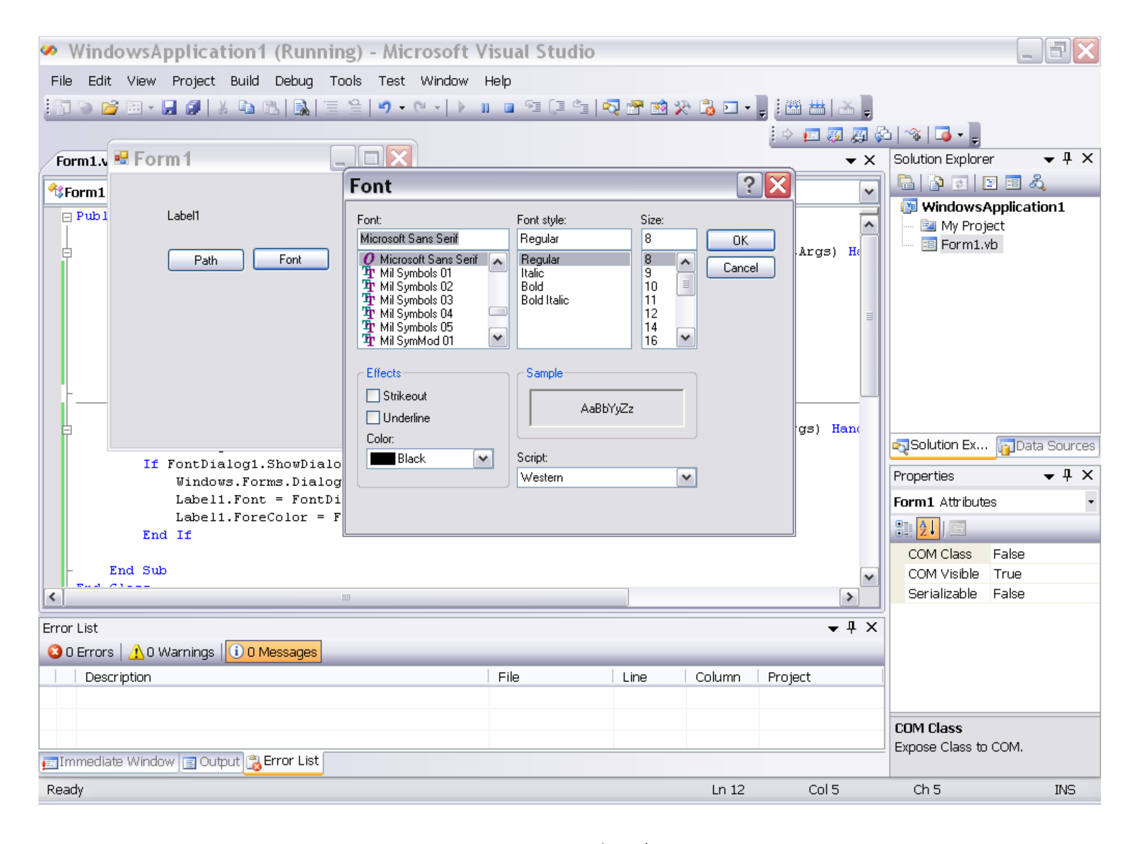
قم بسحب عنصر تحكم من النوع **FontDialog** ورميه فوق النموذج Form1 سيحمل الاسم **FontDialog1** آلياً.

أنقر مرتين فوق المفتاح TextFont لتقوم بفتح محرر التعليمات على الحدث TextFont_Click.

أضف التعليمات التالية للحدث TextFont_Click:

```
FontDialog1.ShowColor = True
If FontDialog1.ShowDialog() = _
   Windows.Forms.DialogResult.OK Then
   Label1.Font = FontDialog1.Font
   Label1.ForeColor = FontDialog1.Color
End If
```

أضرب المفتاح F5 من لوحة المفاتيح لتنفيذ البرنامج. عند تنفيذ البرنامج أنقر المفتاح Font لتظهر نافذة ضبط الخط قم بضبط الخط كما تشاء ثم انقر المفتاح OK ليتغير الخط المستخدم في عنصر التحكم Laberl1.



شكل 4-35: أضافة FontDialog للتطبيق.

1.30.3تدريب: استخدام صندوق الحوار ColorDialog

187.قم بسحب عنصر تحكم من النوع Button ورميه فوق النموذج Form1 ثم قم بتغيير خصائصه للتوافق مع الجدول التالي.

الخاصية	القيمة
Name	FormColor
Text	Color

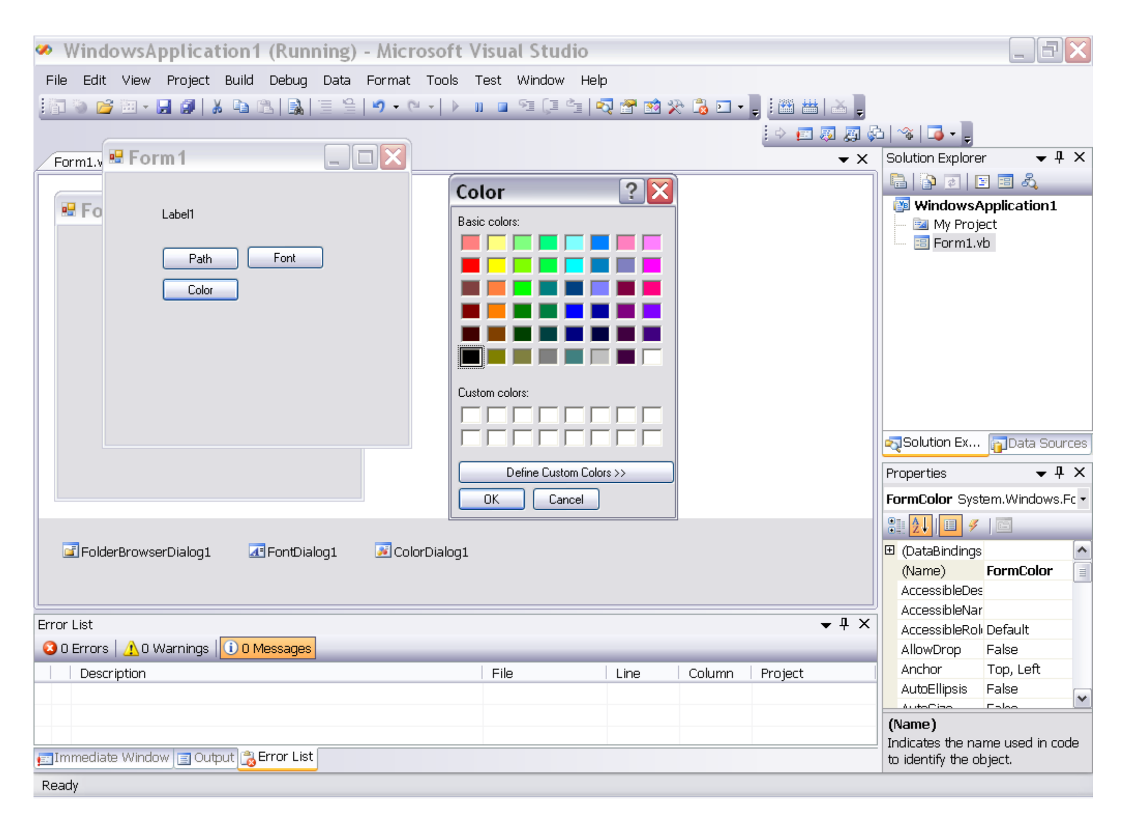
قم بسحب عنصر تحكم من ColorDialog ورميه فوق النموذج Form1 سيحمل الاسم **ColorDialog1** آلياً.

أنقر مرتين فوق المفتاح FormColor لتقوم بفتح محرر التعليمات على الحدث FormColor _Click.

أضف التعليمات التالية للحدث FormColor _Click:

```
If ColorDialog1.ShowDialog() = _
   Windows.Forms.DialogResult.OK Then
   Me.BackColor = ColorDialog1.Color
End If
```

أضرب المفتاح F5 من لوحة المفاتيح لتنفيذ البرنامج. عند تنفيذ البرنامج أنقر المفتاح Color لتظهر نافذة اختيار اللون أختر لون مناسب ثم انقر المفتاح OK ليتغير لون خلفية النموذج.



شكل 4-36: استخدام الـ ColorDialog.

1.31استخدام عنصر تحكم أشرطة الأدوات ToolStrip

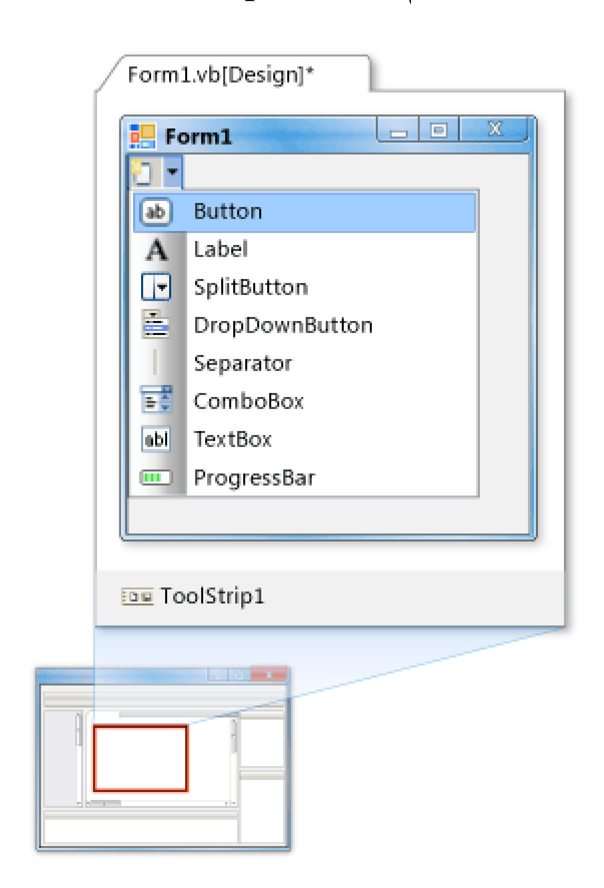
تعتبر أشرطة الأدوات مكون رئيسي من مكونات واجهات التطبيقات العاملة في بيئة التشغيل Microsoft Windows. في هذا القسم سنتعلم كيف نقوم بإضافة أشرطة الأدوات إلى واجهات تطبيقاتنا وكيف نقوم بإضافة مفاتيح الأدوات إليها وربطها بالتعليمات.

1.31.1تدريب: إضافة شريط الأدوات

188. من قائمة File أختار New Project و من تبو يب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

189. قم بسحب عنصر تحكم من النوع ToolStrip من التبويب Menu & Toolbars ورميه فوق النموذج Form1 سيحمل الاسم ToolStrip1 آلياً ويشغل الحرف الأعلى للنموذج Form1.

أنقر فوق السهم لتظهر القائمة المنسدلة و أختر منها Button إضافة مفتاح إلى شريط الأدوات و هو يحمل الاسم **ToolstripButton1**.



شكل 4-37: إضافة Button إلى ToolStrip

قم بضبط خصائص المفتاح **ToolstripButton1 لتصبح كالمبينة في الجدول التالي:**

الخاصية	القيمة
Name	Cut

Text	Cut
DisplayStyle	Text

190.قم بسحب عنصر تحكم من النوع TextBox ورميه فوق النموذج Form1 سيحمل الاسم TextBox1 آلياً.

191.أضبط خصائص الـــ Textbox للتوافق مع الجدول التالي:

الخاصية	القيمة
Size	220,20
Text	This is simple test

أنقر مرتين فوق المفتاح Cut لتقوم بفتح محرر التعليمات على الحدث Cut_ Click.

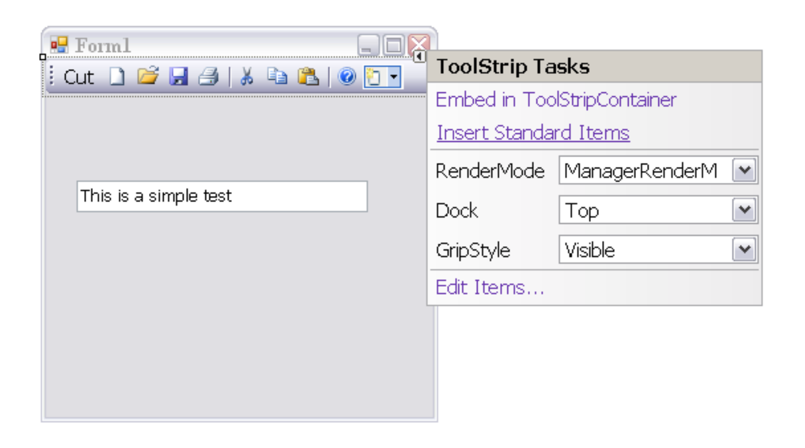
أضف التعليمات التالية للحدث Cut_Click:

```
Me.TextBox1.Cut()
```

أضرب المفتاح F5 من لوحة المفاتيح لتنفيذ البرنامج. ظلل أياً من الكلمات الظاهرة في الـــ TextBox ثم أنقر المفتاح Cut لترى كيف تم قصها.

1.31.2إضافة شريط أدوات قياسي

شر يط الأدوات القيا سي هو شر يط الأدوات الذي يحتوي على وظائف New, Open, Save, Print, Cut, Copy, Paste, Help و هو موجود في غالب تطبيقات Microsoft Windows. ويسهل Visual basic.NET 2008 إضافة مثل هذا الشريط القياسي للتطبيقات، كل ما تحتاجه هو أن تنقر على السهو الموجود في الركن الأيمن الأعلى من شريط الأدوات لتظهر القائمة المنسدلة الخاصة به وأختر منها Insert Standard Items كما في ثم يكون عليك بعد ذلك كتابة التعليمات لكل منها.



4-38: إضافة شريط الأدوات القياسي.

1.32استخدام عنصر التحكم TreeView

يمكنك أن تضع في على بعض تطبيقاتك مظهر مقارب لمظهر Windows Explorer وذلك باستخدام عنصر التحكم TreeView. يعرض عنصر التحكم TreeView البيانات في شكل شجرة تتكون من Nodes. وتنقسم هذه الـ Nodes على ثلاثة أقسام هي:

Parent Nodes

Child Nodes

Root Node

حيث تمثل الـ Node الأساسية التي تتشعب عنها جميع الشعب الـ Root Node، بينما يطلق على أي Node يتشعب منها واحدة او أكثر من الـ Nodes اسم الـ Parent Nodes وأخيراً يطلق اسم الـ Child Nodes على هذه التي لا يتشعب عنها شئ.

1.32.1تدريب: أنشاء متصفح لمواقع الانترنت

192. من قائمة File أختار New Project و من تبويب القوالب Template أختار Windows Application ثم أنقر المفتاح OK.

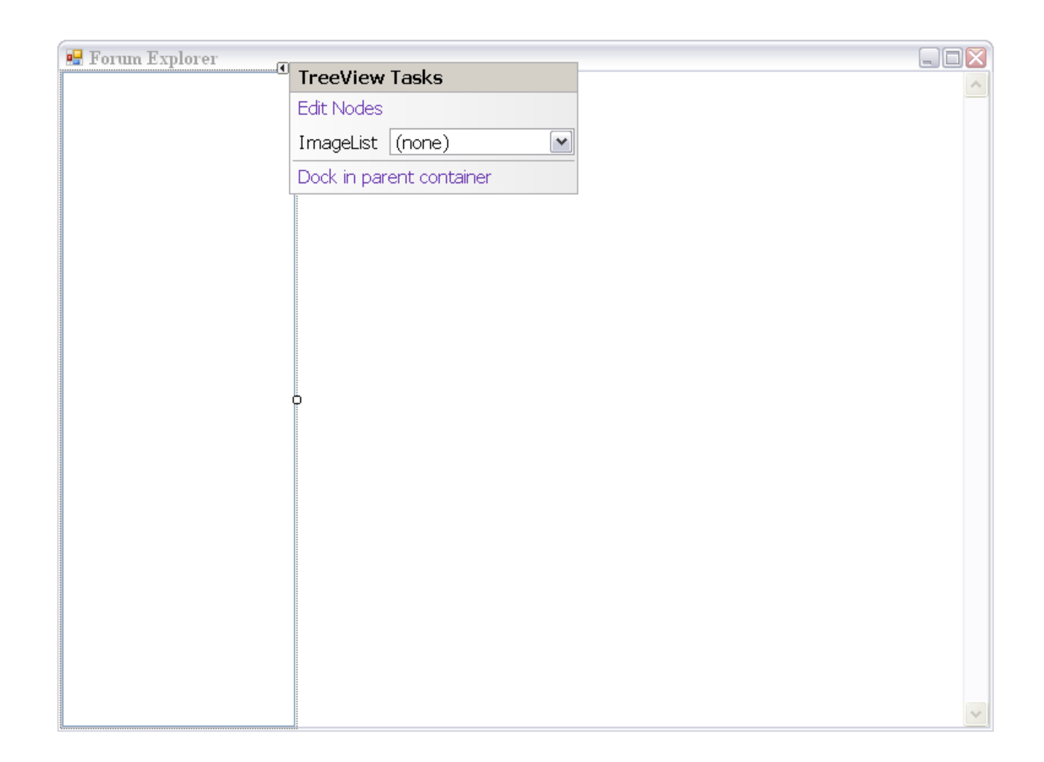
193. قم بسحب عنصر تحكم من النوع TreeView ورميه فوق النموذج Form1 سيحمل الاسم TreeView1 آلياً.

194.قم بسحب عنصر تحكم من النوع WebBrowser ورميه فوق النموذج Form1 سيحمل الاسم WebBrowser1 آلياً.

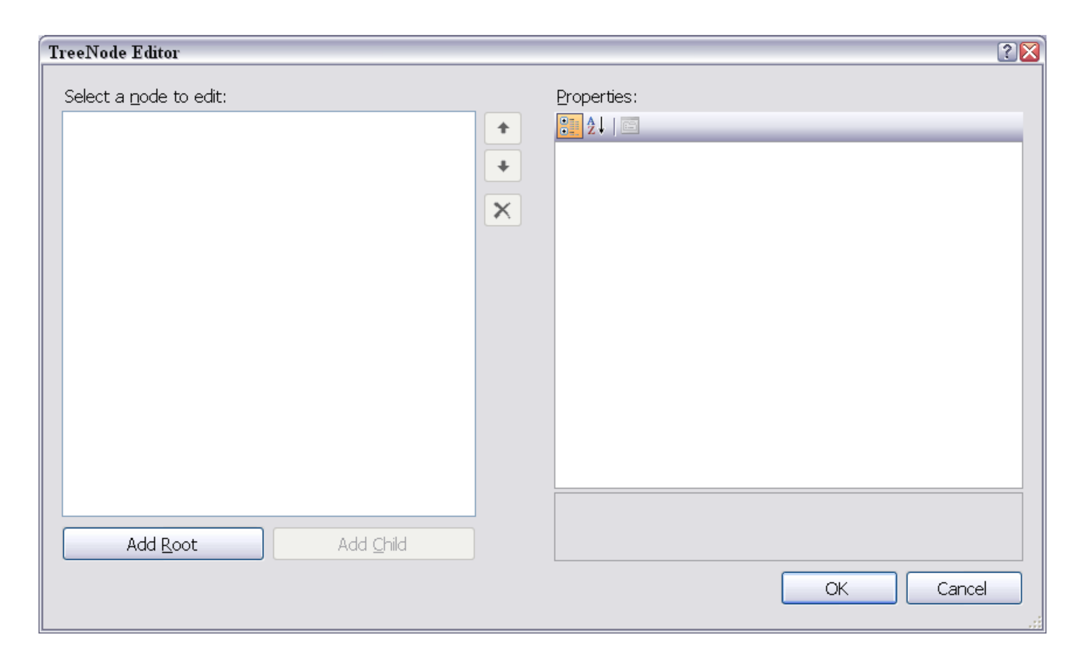
195.قم بضبط خصائص عناصر التحكم للتوافق مع الجدول التالي:

عنصر التحكم	الخاصية	الضبط
Form1	Text	Forum Explorer
	Size	764,564
TreeView1	Dock	Left
	Size	190,530
WebBrowser1	Dock	Fill

قم باختيار عنصر التحكم TreeView1 ثم أنقر على السهم الموجود في الجانب الأيمن من الأعلى لتحصل على القائمة المنسدلة لهذا العنصر وأختر Edit Nodes كما بالشكل شكل 4-39 لتفتح نافذة معالج تحرير الـ Nodes المبينة في شكل 4-40.



شكل 4-39: فتح معالج تحرير الـ Nodes.



شكل 4-40: معالج تحرير الـ Nodes.

أنقر فوق المفتاح Add Root.

في الجا نب الأيمن من المخصص لضبط الخصائص في معالج تحرير الـ Nodes قم بتغيير قيمة الخاصية Text لتصبح Visual Basic Forum.

انقر المفتاح Add Child.

قم بضبط خاصية Text لهذه الـ Node لتصبح Visual Basic Express Edition.

انقر المفتاح Add Child ثانية.

قم بضبط خاصية Text لهذه الـ Node لتصبح Visual Basic IDE.

انقر المفتاح Add Child مرة أخيرة.

قم بضبط خاصية Text لهذه الـ Node لتصبح Visual Basic Language.

أنقر المفتاح OK.

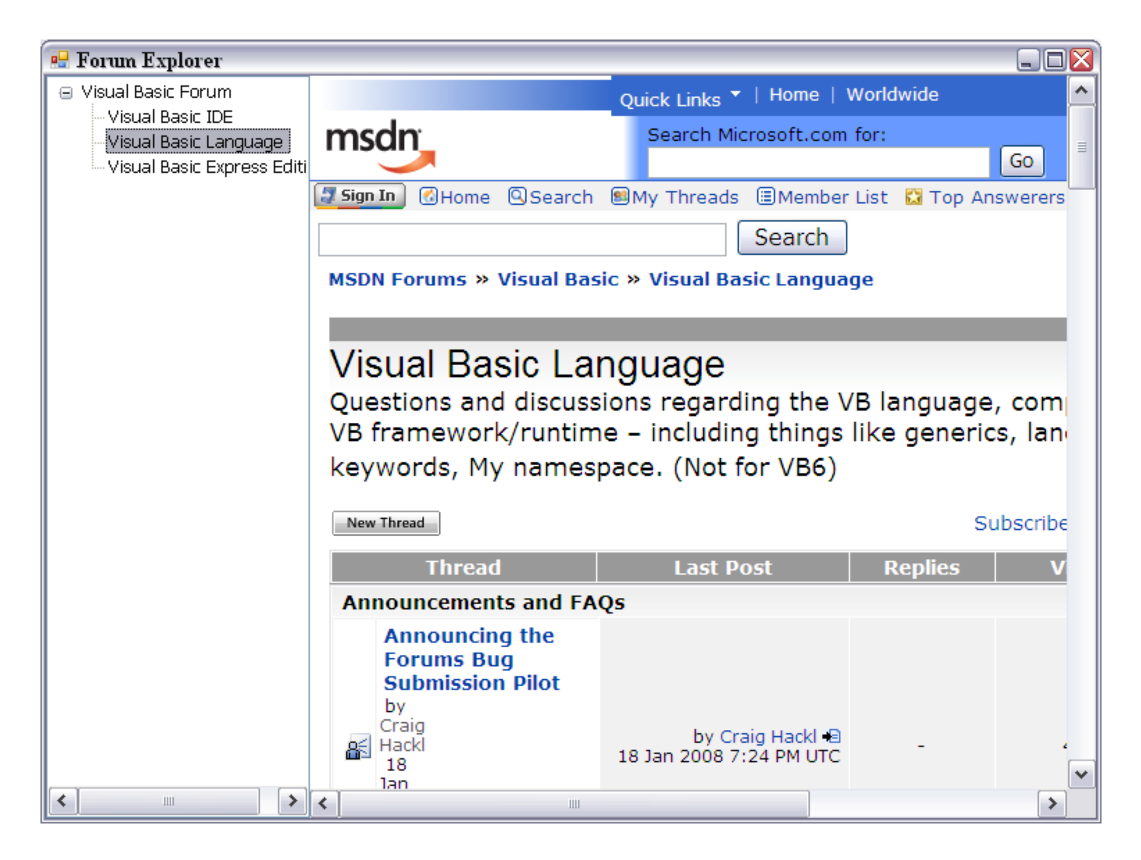
أنقر مرتين فوق TreeView1 لتحصل على معالج الحدث المسمى TreeView1_AfterSelect.

أضف هذه التعليمات للحدث:

```
Select Case e.Node.Text
   Case "Visual Basic Forums"
    WebBrowser1.Navigate("http://go.microsoft.com/" _
        & "fwlink/?LinkID=82999")
   Case "Visual Basic Express Edition"
    WebBrowser1.Navigate("http://go.microsoft.com/" _
        & "fwlink/?LinkID=82994")
   Case "Visual Basic IDE"
    WebBrowser1.Navigate("http://go.microsoft.com/" _
        & "fwlink/?LinkID=82996")
   Case "Visual Basic Language"
    WebBrowser1.Navigate("http://go.microsoft.com/" _
        & "fwlink/?LinkID=82997")
End Select
```

أنقر فوق F5 لتنفيذ البرنامج. لاحظ أنه من المفروض أن تكون متصل بالانترنت حتى لا تحصل على صفحة عدم التحميل.

أختر أي من الـــ Nodes لتحميل محتوياتها في الـــ webBrowser.



شكل 4-41: استخدام البرنامج في تصفح موقع Visual Basic Language.

تقنية Windows Presentation Foundation WPF[1]

في هذا القسم نقدم تقنية Windows Presentation Foundation WPF للقارئ، وتمثل هذه التقنية أحد الإضافات الجديدة[2] إلى Visual Basic.NET 2008 لم تضمها الإصدارات السابقة.

يمكن تعريف تقنية WPF باختصار أنها تقنية لإنتاج الواجهات الرسومية. وهي مضمنة في NET Framework 3.0. ومن ثم فإن التطبيقات التي تحتوي على مثل هذه التقنية يمكن أن تعمل مباشرة على Windows Vista و Windows XP SP2، ولن تعمل على غيرها من النظم.

ولمعرفة الفارق بين الواجهات المطورة باستخدام WPF عن تلك الواجهات التقليدية، يمكن أن تتخيل بين الواجهات التي يجرى تنفيذها باستخدام برنامج Adobe Flash وواجهات التطبيقات المعتادة.

وقد صممت هذه التقنية كي تسمح للمطور بدمجها في تطبيقات Windows التقليدية أو دمجها في تطبيقات الوب، ولذلك فهي تعتمد بصورة كبيرة على تقنية أخرى مخصصة للسكريبت الخاص بالوب هي تقنية Extensible Application Markup Language XAML.

وقد ضمنت Microsoft أدوات لتطوير تطبيقات لهذه التقنية في منتجين لها، Microsoft Visual Studio.NET 2008 و Microsoft Expression Blend. وفي هذا القسم سنتعرف على طرق تطوير التطبيقات باستخدام هذه التقنية مع Visual Basic.NET 2008.

1.33 تصميم واجهة تطبيق باستخدام WPF

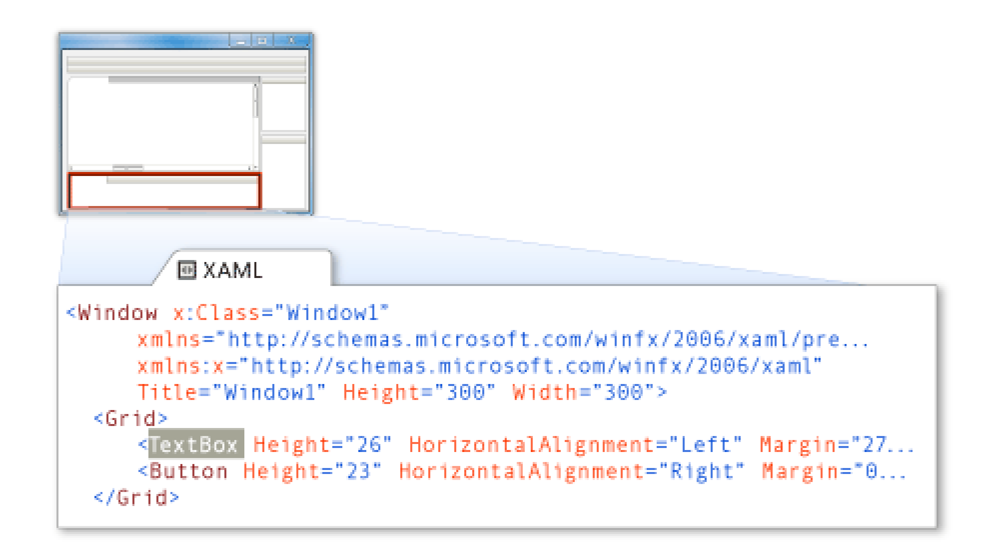
في هذا القسم سنتعرض لإنشاء تطبيق WPF وإضافة عناصر إلى واجهته.

---

[1] عرفت هذه التقنية أثناء مرحلة التجريب باسم Avalon و WinFX.

[2] في المعتاد لا تصل البرمجيات إلى صورة مثلى قبل الإصدارة الثالثة، وفي اعتقادي أن تقنية WPF لم تشذ عن هذه القاعدة.

لا يختلف إنشاء واجهة تطبيق WPF عن أنشاء واجهة تطبيق تقليدي، حيث يمكننا إضافة أي عنصر تحكم بسحبه ورميه فوق الواجهة. لكن هناك بعض الاختلافات التي يجب أن نأخذها في الاعتبار. فإلى جانب نافذة التصميم ونافذة الأدوات ونافذة الخصائص هناك نافذة جديدة تظهر هي نافذة XAML. وكما قدمنا فإن XAML تعني Extensible Application Markup Language والتي هي لغة سكريبت[1] تستخدم لبناء واجهات WPF. الشكل التالي - شكل 5-42 - يبين محرر XAML في بيئة التطوير المتكاملة لـ Visual Basic.NET 2008.



شكل 5-42: محرر XAML

في التطبيقات التقليدية يمكن أن نضيف عناصر التحكم إما عن طريق سحبها ورميها فوق النماذج، أو – وهذه الطريقة غير المعتادة – نستخدم تعليمات لإنشاء عناصر التحكم. وعندما نقوم بسحب عنصر التحكم ورميه فوق النموذج، فإن Visual Studio.NET يقوم تلقائياً بتوليد التعليمات اللازمة لإنشاء هذا العنصر. وبنفس الطريقة عند أنشاء تطبيقات WPF فإن Visual Studio.NET يمكنه أن يولد تعليمات XAML آلياً أو أن يقوم المبرمج بكتابة هذه التعليمات بنفسه.

سكريبت XAML يكون مقسماً إلى وسوم Tags بطريقة هرمية. وكل وسم يكون معرف بين أقواس مثلثة < > كما أن هناك وسوم بدء وأخرى وسوم نهاية. فمثلاً يمكن إضافة عنصر تحكم من النوع Button بمجرد إضافة <Button> </Button> إلى تعليمات

[1] لغات السكريبت هي اللغات التي تعتمد في المقام الأول على الوسوم Tags ولا يتم تجميعها إلا لحظة تنفيذها باستخدام برنامج التنفيذ، وهي شائعة في تقنيات الويب ومن أمثلتها PHP، XML، HTML.

XAML المسئولة عن وصف الواجهة. حيث يمثل <Button> وسم البدء و يمثل </Button> وسم النهاية. كما يمكن ضبط خصائص عنصر التحكم من خلال تعيينها داخل وسوم البدء حيث تكتب قبل قوس الإغلاق > حيث تكتب أسم الخاصية ثم يليها علامة = ثم قيمة الخاصية بين علامات تنصيص ""، فمثلاً <Button Width="200" Height="110"> تعني أنشاء عنصر تحكم من النوع Button بعرض ٢٠٠ نقطة وارتفاع ١١٠ نقطة.

وعندما يقوم المبرمج بسحب عنصر تحكم من الـ Toolbox ورميه فوق النموذج فإن Visual Basic يقوم بإنشاء وسوم XAML الخاصة بهذا العنصر. فمثلاً عند سحب عنصر التحكم Button ورميه فوق النموذج تضاف الوسوم التالية إلى تعليمات XAML.

```
<Button Height="23" Margin="102,45,101,0"
Name="Button1" VerticalAlignment="Top">Button</Button>
```

وكما يرى القارئ تتعين خصائص هذا العنصر من خلال هذه الوسوم.

1.33.1 تدريب: أنشاء تطبيق WPF

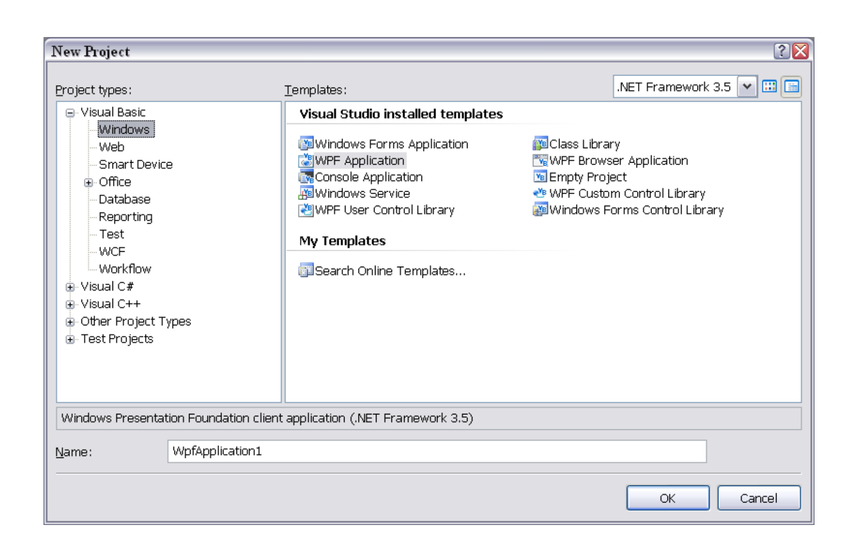
196. من قائمة File أختار New Project. تنفتح النافذة New Project.

197. تأكد أن إصداره .NET Framework المستخدمة هي على الأقل الثالثة.

198. من Templates أختار WPF Applications.

199. في الخانة Name أكتب اسم التطبيق الذي ترغب في إنشاءه أو دع التسمية التلقائية wpfApplication1.

200. أنقر المفتاح OK.



شكل 5-43: أنشاء تطبيق WPF جديد

يتم أنشاء تطبيق WPF جديد، يضم نافذة جديدة أسمها Window1، وتظهر وسوم XAML الخاصة بالتطبيق في محرر XAML كما يلي:

```
<Window x:Class="Window1"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Title="Window1" Height="300" Width="300">
   <Grid>

   </Grid>
</Window>
```

أنقر فوق Window1 لاختيارها.

غ ير اخ صية Title في و سوم XAML لتصبح Title=”WPF Application” وتصبح وسوم XAML كالتالي:

```
<Window x:Class="Window1"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xam
l"
    Title="WPF Application" Height="300" Width="300">
    <Grid>

    </Grid>
</Window>
```

يتغير عنوان النافذة ليصبح WPF Application يمكن أن يختبر القارئ أن يقوم بتغيير الخصائص الأخرى مثل Width و Height.

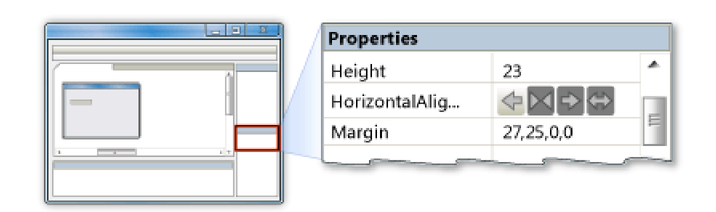
1.33.2تدريب: إضافة عناصر التحكم إلى نافذة WPF.

في هذا التدريب سنقوم بإضافة عناصر تحكم إلى تطبيق WPF. و هو ما سنقوم به بالطريقة التقليدية بأن نقوم بسحبه من الـ Toolbox ورميه فوق النموذج. ثم سنقوم بضبط بعض خصائص هذه العناصر باستخدام وسوم XAML.

قم بسحب TextBox من الــ Toolbox ثم أرميه فوق نافذة التطبيق.

أنقر فوق TextBox لتختاره.

في نافذة الخصائص ابحث عن الخاصية HorizontalAlignment ثم أجعل قيمتها Left.



شكل 5-44: ضبط الخاصية HorizontalAlignment

أضبط بقية الخصائص لتوافق الجدول التالي:

الخاصية	الضبط
VerticalAlignment	Top
Width	75
Heigh	26

مستخدما وسوم XAML قم بتحرير خاصية Width الخاصة بعنصر التحكم TextBox لتصبح ١٤٠ و قم بتغيير خاصية Margin لتصبح 30,56,0,0. من المفترض أن تكون الوسوم لها الصيغة التالية:

```
<TextBox Height="26" HorizontalAlignment="Left"
Margin="30,56,0,0"
   Name="TextBox1" VerticalAlignment="Top"
Width="140" />
```

أضف Button إلى واجهة WPF.

قم بتعديل وسوم XAML الخاصة بهذا العنصر لتصبح كالمبنية فيما يلي:

```
<Button Height="23" HorizontalAlignment="Right"
Margin="0,59,35,0"
   Name="Button1" VerticalAlignment="Top"
   Width="75">Submit</Button>
```

انقر F5 لتنفيذ التطبيق.

1.34عناصر تحكم WPF الشائعة

في هذا القسم نتناول عدد من عناصر التحكم الخاصة بواجهات WPF الأكثر استخداما. يساعد WPF كما قدمنا في إنشاء واجهات استخدام محسنة الشكل، وحتى عناصر التحكم التي استخدمت من قبل في التطبيقات التقليدية يمكن تحسين مظهرها في WPF.

الطريقة الأبسط لإضافة عناصر التحكم إلى واجهات WPF هو عن طريق سحبها ورميها من Toolbox إلى النماذج كما قمنا في التدريبات السابقة، و في هذه الحالة تظهر عناصر التحكم في صورتها التقليدية. كما يمكن للمبرمج تغيير مظهر هذه العناصر من خلال تغيير خصائصها سواء من نافذة الخصائص أو من محرر XAML[1].

1.34.1تدريب: إضافة عنصر تحكم لتطبيق WPF وربطه بالتعليمات

201.من قائمة File أختار New Project. تنفتح النافذة New Project.

202.تأكد أن إصداره .NET Framework المستخدمة هي على الأقل الثالثة.

[1] تنصح Microsoft المبرمجين باستخدام البرنامج Microsoft Blend لتطوير وسوم XAML للحصول على مظهر جذاب لعناصر التحكم، حيث أن تطوير الوسوم في Visual Studio.NET 2008 يعتبر صعباً مقارنة بـ Microsoft Blend.

203.من Templates أختار WPF Applications.

204.في الخانة Name أكتب اسم التطبيق الذي ترغب في إنشاءه أو دع التسمية التلقائية wpfApplication1.

205.أنقر المفتاح OK.

206.قم بسحب TextBox من الـــ Toolbox ثم أرميه فوق نافذة التطبيق.

207.أنقر فوق TextBox لتختاره.

208.أضبط خصائص عنصر التحكم TextBox لتوافق الجدول التالي:

الخاصية	الضبط
VerticalAlignment	Top
Width	75
Heigh	26

أضف Button إلى واجهة WPF.

قم بتعديل وسوم XAML الخاصة بهذا العنصر لتصبح كالمبنية فيما يلي:

```
<Button Height="23" HorizontalAlignment="Right"
Margin="0,59,35,0"
   Name="Button1" VerticalAlignment="Top"
   Width="75">Add</Button>
```

أضف ListBox إلى واجهة WPF.

أنقر مرتين فوق Button1 لتفتح نافذة تحرير التعليمات.

أضف التعليمات التالية للحدث.

```
If TextBox1.Text IsNot "" Then
   ListBox1.Items.Add(TextBox1.Text)
   TextBox1.Text = ""
End If
```

أنقر F5 لتشغيل البرنامج، قم بكتابة أي نص في صندوق النصوص ثم أنقر فوق المفتاح Add لإضافته إلى القائمة.

1.34.2قائمة بعناصر تحكم WPF

فيما يلي قائمة بأهم عناصر تحكم WPF ووظائفها.

عنصر التحكم	وصفه
System.Windows.Controls.Border	يعرض إطار حول المحتويات
System.Windows.Controls.Button	مفتاح
System.Windows.Controls.CheckBox	صندوق تاشير
System.Windows.Controls.ComboBox	صندوق قائمة مدمجة
System.Windows.Controls.Grid	منطقة الشبكة التي يضيف فوقها المستخدم عناصر التحكم
System.Windows.Controls.Image	يعرض صور
System.Windows.Controls.Label	عرض العناوين Labels
System.Windows.Controls.ListBox	صندوق قائمة
System.Windows.Controls.RadioButto n	مفتاح راديو
System.Windows.Control.TabControl	يسمح بتنفيذ تبيوبات عدة
System.Windows.Controls.TextBox	صندوق النصوص

1.35أنشاء معالج حدث لعناصر تحكم WPF

في هذا القسم نتناول التعامل مع معالج أحداث Event Handler عناصر التحكم WPF.

يتشابه أنشاء معالج الحدث لعناصر تحكم WPF مع أنشاء معالج حدث عناصر التحكم التقليدية. لكن الذي يتميز به أنشاء معالج الحدث لعناصر تحكم WPF هو أمكان الدمج بين تعليمات Visual Basic ووسوم XAML لتصميم معالج الحدث. ويتم ذلك بان يضاف أسم الحدث والطريقة المستخدمة لخصائص عنصر التحكم كما تبدو في وسوم XAML ثم إضافة التعليمات باستخدام محرر التعليمات.

1.35.1تدريب: أنشاء معالج حدث لعنصر تحكم من النوع Button

209.من قائمة File أختار New Project. تنفتح النافذة New Project.

210.تأكد أن إصداره .NET Framework المستخدمة هي على الأقل الثالثة.

211.من Templates أختار WPF Applications.

212.في الخانة Name أكتب اسم التطبيق الذي ترغب في إنشاءه أو دع التسمية التلقائية .wpfApplication1

213.أنقر المفتاح OK.

214.أضف Button إلى واجهة WPF.

215.أنقر مرتين فوق Button1.

216.أضف التعليمات التالية:

```
MsgBox("Event handler was created by double-clicking
the button.")
```

217.أضف Button ثاني إلى واجهة WPF.

218.قم بتغيير وسوم XAML الخاصة بالمفتاح Button2 لتصبح كالتالي:

```
<Button Height="23" Margin="67,96,136,0"
     Name="Button2"
     VerticalAlignment="Top"
  Click="ButtonOkClicked">Button</Button>
```

حيث Click أسم الحدث و ButtonOKClicked هو اسم معالج الحدث الذي سوف نقوم بتصميمه.

أنتقل إلى محرر التعليمات وأضف التعليمات التالية عقب Class Window1 مباشرة.

```
Sub ButtonOKClicked(ByVal Sender As Object, _
  ByVal e As RoutedEventArgs)
    MsgBox("Event handler was created manually.")
End Sub
```

أنقر F5 لتنفيذ التطبيق.

1.36أنشاء تطبيق WPF للرسم

سنتعلم في هذا القسم الخطوات اللازمة لإنشاء تطبيق WPF يساعد في رسم الصور.

219.من قائمة File أختار New Project. تنفتح النافذة New Project.

220.تأكد أن إصداره .NET Framework المستخدمة هي على الأقل الثالثة.

221.من Templates أختار WPF Applications.

222.في الخانة Name أكتب اسم التطبيق Ink Pad.

223.أنقر المفتاح OK.

224.انقر فوق النافذة Window1 لتختارها.

225.في نافذة الخصائص قم بضبط خصائص Window1 لتصبح.

الخاصية	الضبط
Width	٣٧٠
Height	٥٥٠
Title	Ink Pad
Background	**Brown**

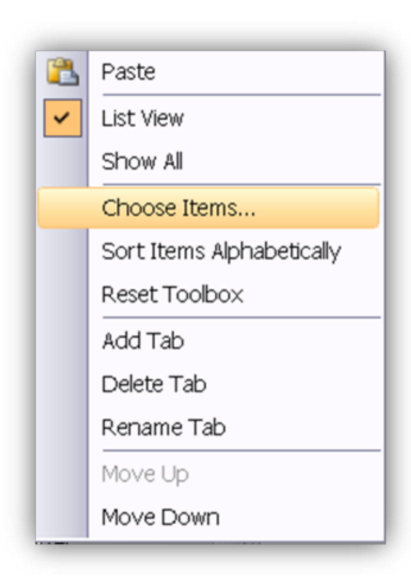
أنقر مفتاح الفارة اليمين فوق أي مكان فوق الـ Toolbox لتظهر القائمة المنسدلة الخاصة به والمبينة في شكل 5-45.

أختار Choose Items لتظهر النافذة المبينة في شكل 5-46.

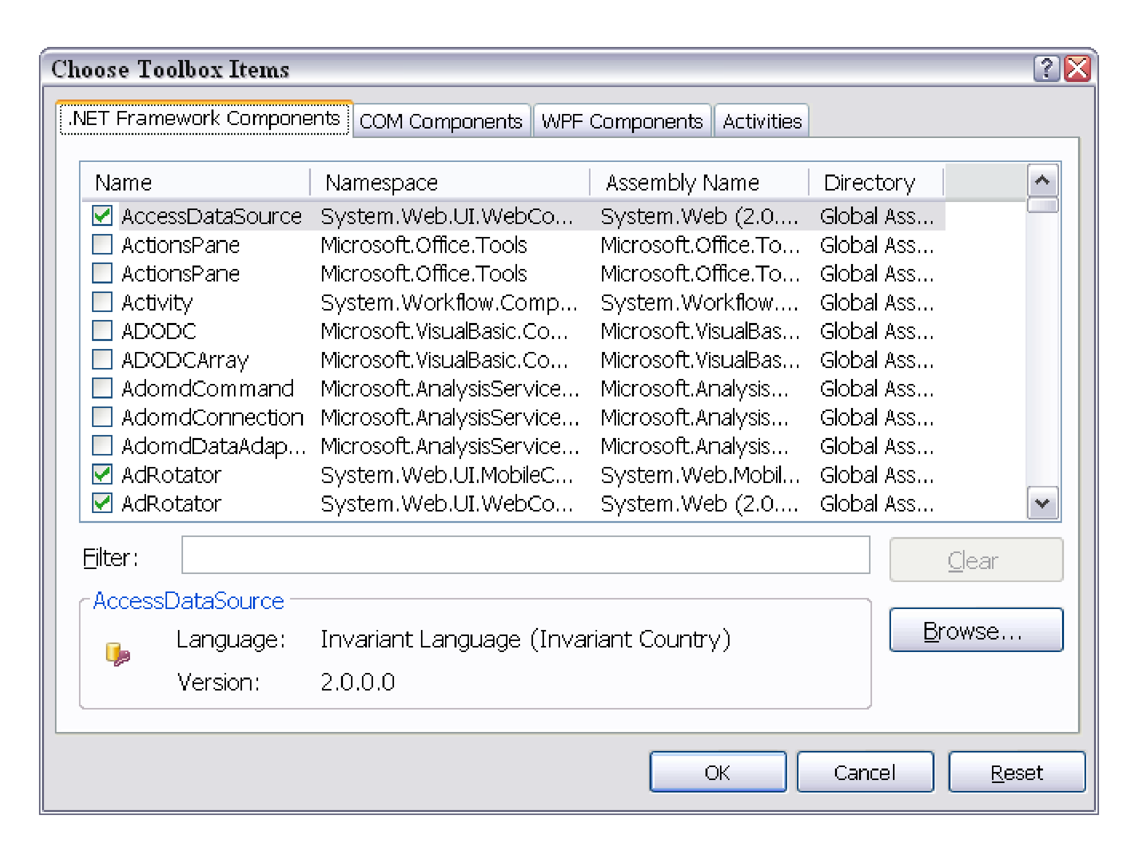
انقر فوق التبويب WPF Components.

أبحث عن InkCanvas ثم أنقر على المربع المقابل له.

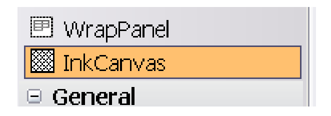
انقر المفتاح OK لتظهر أيقونة InkCanvas في الـ Toolbox كما هو مبين في شكل 5-47.



شكل 5-45: أنقر Choose Items



شكل 5-46: نافذة Choose Toolbox Items



شكل 5-47: أيقونة InkCanvas

قم بسحب الـــ InkCanvas من Toolbox ورميها فوق النافذة.

مسـ تخدماً نافذة Properties قم بضـ بط خصـ ائص InkCanvas لتصـ بح كالتالي:

الخاصية	الضبط
Width	Auto
Height	Auto
HorizontalAlignment	Stretch
VerticalAlignment	Stretch
Margins	**9,9,9,68**
Background	**Light Yellow**

قم بإضا فة مفتاح ين ت حت InkCanvas أجـ عل button1 إلى اليسـ ار و Button2 على اليمين.

غير خصائص Button1 مستخدماً محرر XAML لتصبح كالتالي:

```
<Button Height="23" HorizontalAlignment="Left"
Margin="85,0,0,24"
   Name="Button1" VerticalAlignment="Bottom"
   Width="75">Clear</Button>
```

غير خصائص Button2 مستخدماً محرر XAML لتصبح كالتالي:

```
<Button Height="23" HorizontalAlignment="Right"
Margin="0,0,72,24"
   Name="Button2" VerticalAlignment="Bottom"
   Width="75">Close</Button>
```

أنقر مرتين فوق المفتاح Clear وأضف التعليمات التالية إلى الحدث

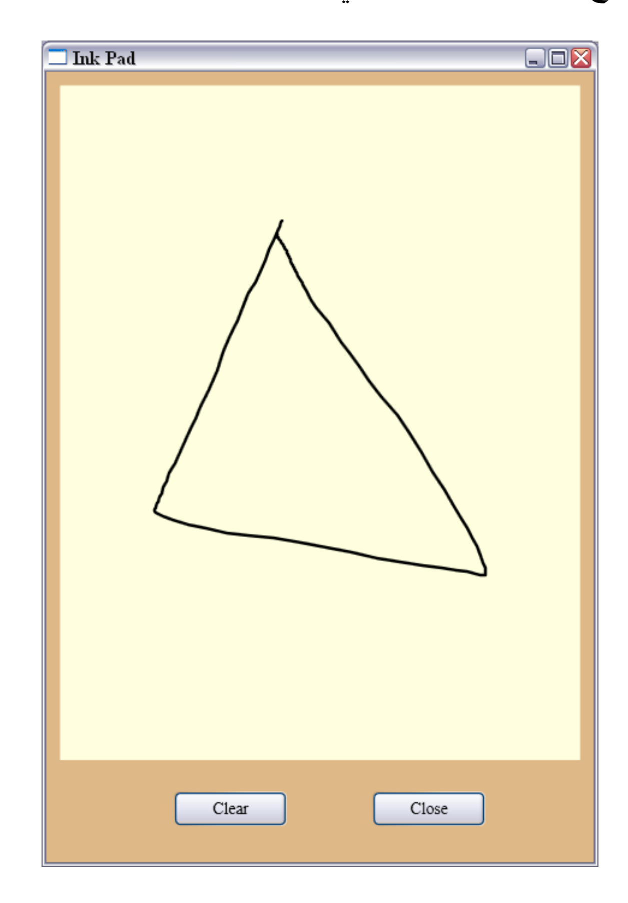
```
Me.InkCanvas1.Strokes.Clear()
```

أنقر مرتين فوق المفتاح Close وأضف التعليمات التالية إلى الحدث

```
Me.Close()
```

أنقر المفتاح F5 لتنفيذ البرنامج.

مستخدماً مؤشر الفأرة قم بالرسم فوق InkCanvas وإذا أحسست أن الرسم غير جميل قم بنقر مفتاح Clear وعندما تنتهي أنقر Close.



شكل 5-48: التطبيق Ink Pad أثناء التشغيل.

معالجة الأخطاء

عندما تقوم بكتابة برنامج حاسوب، فإن وقوع الأخطاء من الأمور الواردة. يمكن أن تقوم بكتابة بعض العبارات بصورة خاطئة لا يتوقعها الحاسوب، أو أن تكون هناك أخطاء مخفية تتسبب في توقف البرنامج، أو الأسوأ تتسبب في أن يقوم البرنامج بتقديم نتائج خاطئة. عندما يكون برنامجك يحتوي على أخطاء يجب عليك العثور على هذه الأخطاء ومعالجتها، هذه العملية التي يطلق عليها أسم إزالة الأخطاء Debugging.

1.37البحث عن الأخطاء

فيما يلي، سنتعلم كيف نقوم بعملية إصلاح البرنامج من خلال إزالة الأخطاء.

بغض النظر عن الطريقة التي قمت بها بتصميم برنامجك أو كتابة شفرته، فإن الأخطاء من الأمور الواردة. بعض الأخطاء تمنع البرنامج من أن يبدأ بعضها الأخر يتسبب في انهيار crash البرنامج وأسوء من هذا كله أن يعمل البرنامج ولكن يعطي نتائج خاطئة.

بالطبع عندما تقع الأخطاء، فمن الضروري أن تعثر عليها وتقوم بإصلاحها. يطلق على أخطاء البرامج عامة الاسم bug كما يطلق على عملية إزالة هذه الأخطاء الاسم debugging.

عملية إزالة الأخطاء هي عملية تكرارية Iterative، فهي عملية ستقوم بتكرارها مرة بعد مرة أثناء عملية البرمجة. فعندما تكوم بكتابة جزء من الشفرة ستقوم باختبار البرنامج، إذا حدث خطأ ستقوم بالبحث عنه وإصلاحه ثم تقوم بتشغيل البرنامج مرة أخرى، وتتكرر هذه العملية كل مرة تجد فيها خطا.

في معظم الحالات لن ترغب في إيقاف تنفيذ البرنامج لتقوم بالبحث عن الخطأ وإصلاحه. بل ربما كان الأفضل أن تقوم بالعثور على الخطأ وقت ظهوره ومعالجته ثم تدع البرنامج يستمر في التشغيل و هي العملية التي تسمى التحديث والاستمرار Edit and Continue.

تقوم بيئة التطوير المتكاملة لـ Visual Basic – Visual Basic IDE – بعمل عمليات إزالة الأخطاء، حيث تحتوي على العديد من الأوامر والنوافذ التي تستخدم في البحث عن الأخطاء. هذه العناصر التي سنتناولها في أجزاء هذا القسم.

1.37.1 تدريب: استخدام خاصية Edit & Continue

المثال الذي سوف نعرضه فيما يلي يتضمن ما يسمى بالاستثناء exception. الاستثناءات هي كائنات يتم إنشاءها عندما يشعر البرنامج بأن هناك خطأ قد وقع في التشغيل. هناك أنواع مختلفة من الاستثناءات التي يتم إنشاءها، وتعتمد هذه الأنواع على نوع الخطأ نفسه. في الوضع العادي، إذا حدث أي استثناء يظهر صندوق حوار ليوضح نوع الخطأ ويساعد في إصلاحه.

226. من القائمة File أنقر فوق New Project، من نافذة New Project أختار Template ومنها Windows Application ثم أنقر المفتاح Ok.

أنقر مرتين فوق النموذج ليظهر الــ Code Editor.

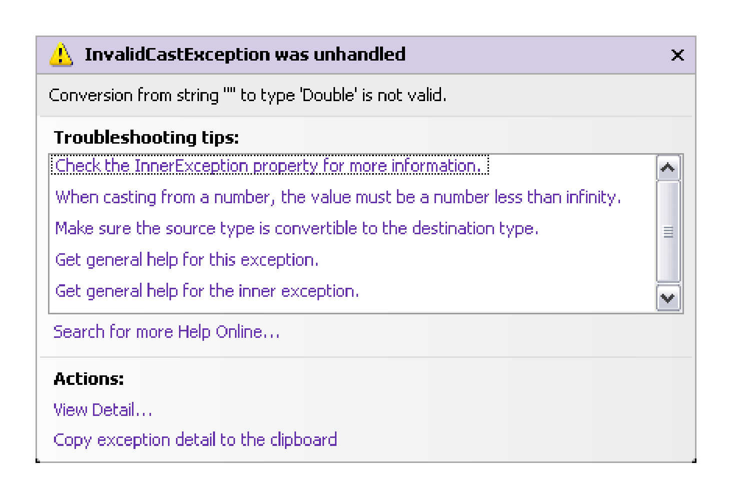
قم بكتابة الشفرة التالية في عامل الحدث Form_Load:

```
Dim number As Integer = 1
Dim numbers As String = ""
MsgBox(numbers + 1)
```

أضرب المفتاح F5 لتشغيل البرنامج. هذا البرنامج سوف يتعطل وتظهر نافذة الاستثناء المبينة في شكل 6-49. الاستثناء حدث نتيجة حدوث خطأ كتابي Typographical error حيث قام المبرمج بكتابة المتغير النصي numbers محل المتغير الصحيح number داخل وظيفة MsgBox. لاحظ أن البرنامج لم ينقطع عن العمل، فقط هو توقف عن العمل منتظراً أن تقوم بإصلاح الخطأ وهو ما يمثل خاصية Edit and Continue.

في نافذة Code Editor قم باستبدال numbers+1 بــ number+1.

أضرب F5 ليستمر البرنامج في العمل.



شكل 6-49: نافذة الاستثناء

1.38 أنواع الأخطاء

فيما يلي نتعرف على أنواع الأخطاء التي تحدث عن كتابة البرامج.

حتى هؤلاء المبرمجين المحترفين يقومون بالأخطاء، لكنهم يعرفون كيف يتعرفون على أخطاء هم وكيف يمكنهم إزالتها. أنهم ينظرون إلى الأخطاء كجزء من عملية البرمجة. قبل أن نقوم بتعلم كيف نقوم بعملية إزالة الأخطاء debugging سيكون من الأفيد أن نتعرف على أنواع الأخطاء الممكن وقوعها.

تنقسم الأخطاء الحادثة أثناء عملية البرمجة إلى ثلاثة أنواع، أخطاء التجميع Compilation Error وأخطاء التشغيل Run-time Error والأخطاء المنطقية Logic Error.

1.38.1 أخطاء التجميع Compilation Errors

أخطاء التجميع – والتي يطلق عليها أيضاً أسم أخطاء المجمع Compiler Error – هي تلك الأخطاء التي تمنع برنامج من العمل. عندما تقوم بضرب المفتاح F5 يقوم Visual Basic بتجميع شفرة برنامج إلى اللغة الثنائية binary Language التي يفهمها الحاسوب. لو أن مجمع Visual Basic صادف شفرة لا يعرفها فإنه يقع في خطأ التجميع و من ثم يتوقف عن تجميع البرنامج وبالتالي لا يعمل البرنامج.

معظم أخطاء البرمجة تقع نتيجة أخطاء في كتابة الشفرة. مثل أن يقوم المبرمج بكتابة أي من الكلمات المحجوزة بطريقة خاطئة أو ينسى وضع علامات التنصيص أو علامات الفاصلة أو مثلاً كتابة End If بدون كتابة If قبلها.

من حسن الحظ، أن Visual Basic مصمم بحيث يكتشف معظم هذه الأخطاء أثناء تحرير البرنامج نفسه وهو ما ستتعلمه في جزء لاحق من هذا الفصل.

1.38.2أخطاء التشغيل Run-time Errors

أخطاء التشغيل – كما هو واضح من أسمها – تحدث أثناء تشغيل البرنامج. وعادة ما يكون سبب هذه الأخطاء العمليات التي لا يمكن للحاسوب أن يقوم بها. من أكثر هذه العمليات التي تسبب أخطاء التشغيل شيوعاً هو القسمة على الصفر.

عند وقوع مثل هذا النوع من الأخطاء يمكن، يمكنك استخدام أدوات إزالة الأخطاء التي تقدمها Visual Basic حتى يتسنى أن تعثر على هذه الأخطاء وتقوم بمعالجتها.

1.38.3الأخطاء المنطقية Logic Error

الأخطاء المنطقية هي أكثر أنواع أخطاء البرمجة خطراً، فهذه الأخطاء ينشأ عنها خطأ في سريان البيانات داخل البرنامج ومن ثم ظهور نتائج خاطئة غير متوقعة عن العلميات التي يقوم بها الحاسوب. ومن جانب آخر هي بمثابة أخطاء خفية بحيث يصعب على أدوات إزالة الأخطاء في Visual Basic التعرف عليها. ومن ثم فإن هناك عامل كبير على المبرمج في مراجعة برنامجه ليتعرف على مكان الخطأ المنطقي.

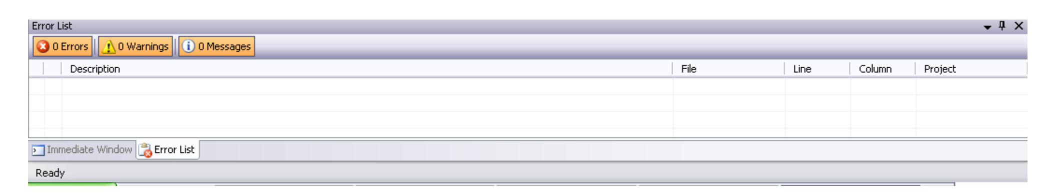
1.39العثور على أخطاء التجميع وإصلاحها

فيما يلي نتعلم كيفية العثور على أخطاء التجميع وإصلاحها.

كما تعلمنا فيما سبق أن أخطاء التجميع تنشأ عن وجود شفرة غير مفهومة بالنسبة لمجمع Visual Basic. ولأن هذا النوع من الأخطاء يعوق البرنامج عن بدأ التشغيل، فمن الأفضل العثور على هذه الأخطاء وعلاجها قبل الشروع في تشغيل البرنامج.

العثور على هذا النوع من الأخطاء لحسن الحظ بسيط. عندما تضرب المفتاح F5، ويتعثر المجمع في أي من هذه الأخطاء، تظهر على الشاشة صندوق حواري يحمل العبارة There were build errors. Continue؟ إذا اخترت Yes فإنه سوف يتم تشغيل آخر نسخة من برنامج قبل أن تقوم بكتابة الشفرة الخاطئة، أما إذا اخترت No فإن البرنامج سوف يتوقف عن العمل وتظهر نافذة Error List.

نافذة Error List تقوم بعرض معلومات حول أخطاء التجميع، تضم هذه النافذة المبينة في الشكل اللاحق وصف الخطأ وموقعه من الشفرة. لو نقرت نقراً مزدوجاً فوق أي من الأخطاء المبينة في نافذة Error List فإن السطر الذي يحتوي على الخطأ يتم تظليله. يمكن نك ضرب المفتاح F1 لي تم عرض وثائق المسا اعدة Help Documentation لـ Visual Basic للتعرف على معلومات أكثر حول الخطأ وكيفية إصلاحه.



شكل 6-50: نافذة Error List

وعادة ما يقوم محرر الشفرة لـ Visual Basic Code Editor – Visual Basic – بتعيين قسم كبير من أخطاء التجميع أثناء تحر ير شفرة البرنا مج من خلال الخاصية المسماة IntelliSense. فعندما تقوم بكتابة أي كلمة من الكلمات المحجوزة أو الوظائف أو بارامترات الوظائف يقوم الـ IntelliSense بإظهار قائ مة تحتوي على جم يع الكلمات المحجوزة أو الوظائف أو بارامترات الوظائف المشابهة لتنتقي منها ما يناسبك استخدامك.

1.39.1 تدريب: العثور على أخطاء التجميع وإصلاحها

227. من القائ مة File أنقر فوق New Project، من نافذة New Project أختار Template ومنها Windows Application ثم أنقر المفتاح Ok.

أنقر مرتين فوق النموذج ليظهر الـــ Code Editor.

قم بكتابة الشفرة التالية في عامل الحدث Form_Load:

```
End If
```

عندما تضرب المفتاح Enter يظهر خط أزرق متموج تحت العبارة End If.

قم بتغيير هذه الشفرة لتصبح كما الآتي

```
If 1 < 2 Then
End If
```

تلاحظ أن الخط الأزرق المتموج اختفى. قم بإضافة الشفرة التالية بعد بين If و End If.

```
MgBox("Hello")
```

أضرب المفتاح F5. تظهر العبارة There were build errors. Would you like to continue and run the last successful build?

أختار No تظهر نافذة Error List تبين أن الخطأ هو Name 'MgBox' is not declared.

أنقر نقراً مزدوجاً فوق الخطأ ثم قم بتعديل العبارة الخاطئة لتصبح MsgBox("Hello").

أضرب المفتاح F5 مرة أخرى وأنظر هل يعمل البرنامج.

1.40 العثور على أخطاء التشغيل وإصلاحها

فيما يلي نتعلم كيف يمكن أن نعثر على أخطاء التشغيل ونقوم بإصلاحها.

كما تعلمنا سابقاً أن أخطاء التشغيل تنجم عن عمليات غير مسموح بها. عند وقوع أي من هذه الأخطاء يتوقف البرنامج عن العمل وتظهر رسالة خطأ حتى يمكنك التعرف على وقوع خطأ تشغيل وتقوم بإصلاحه.

معظم الأخطاء التشغيلية تقع نتيجة لخطأ في الشفرة نفسها، كأن تنسى أن تقوم بتخصيص قيمة لمتغير قبل أن تقوم باستخدامه. عند تشغيل برنامج ووقوع أخطاء تشغيل، يتوقف البرنامج عن العمل ويظهر الصندوق الحواري مساعد الاستثناء Exception Assistant في نافذة Code Editor. عندما يحدث هذا فإن البرنامج يدخل حالة تسمى Break Mode حيث يمكنك أن تقوم بعمليات إزالة الأخطاء.

صندوق الحوار مساعد الاستثناء Exception Assistant يحتوي على وصف للخطأ ونصائح حول سبب وقوع الخطأ. بالنقر على هذه النافذة تظهر وثائق المساعدة التي تحتوي على المزيد من التعليمات القيمة.

1.40.1تدريب: العثور على أخطاء التشغيل وإصلاحها

228. من القائمة File أنقر فوق New Project، من نافذة New Project أختار Template ومنها Windows Application ثم أنقر المفتاح Ok.

أنقر مرتين فوق النموذج ليظهر الـــ Code Editor.

قم بكتابة الشفرة التالية في عامل الحدث Form_Load:

```
Dim miles As Integer = 0
Dim hours As Integer = 0
Dim speed As Integer = 0
miles = 55
speed = miles / hours
MsgBox(CStr(speed) & " miles per hour")
```

أضرب المفتاح F5 لتنفيذ البرنامج. تظهر النافذة مساعد الاستثناء Exception Assistant تحمل الرسالة Overflow Exception was unhandled كما يظهر خط متقطع يربط ما بين النافذة مساعد الاستثناء Exception Assistant ومكان الخطأ من البرنامج.

ضع مؤشر الفأرة فوق المتغير miles وأتركه لثواني سوف تظهر نافذة صفراء صغيرة – نافذة النصح tool tip window – تحمل فقط 55 miles.

كرر نفس الخطوة ٥ مع استبدال miles بـــ hours تلاحظ أن الرسالة أصبحت 0 hour. وحيث أن القسمة على صفر غير معترف فيها في الكومبيوتر فإن الخطأ تولد هنا.

أضف السطر التالي بعد سطر miles = 55:

```
hours = 2
```

أعد تنفيذ البرنامج وأنظر إن كان سوف يعمل أم لا.

1.41استخدام النافذة الوسيطة Intermediate Window

في هذا القسم نستكشف إمكانات اختبار التعليمات باستخدام النافذة الوسيطة Intermediate Window. ففي القسم السابق عرفنا كيف يمكن الاستدلال على الأخطاء باستخدام مساعد الاستثناء Exception Assistant لكن في بعض الأحيان لا يكون من الواضح أين الخطأ ويستحيل على المبرمج أن يعرف مكان الخطأ بدون أن يقوم بتعديل التعليمات. في هذه الحالة تكون الأداة المثلى لمعالجة الأخطاء هي النافذة الوسيطة Intermediate Window.

عندما يكون البرنامج في الـ Break Mode، فإن النافذة الوسيطة يمكن ان تستخدم لتنفيذ قسم من التعليمات وتقييم المتغيرات والتعبيرات. على سبيل المثال، إذا تولد خطأ في أثناء التشغيل run-time بسبب متغير فارغ (لم تخصص له قيمة)، فإن المبرمج يريد أن يتعرف على قيمة المتغير، وهو ما تسمح به النافذة الوسيطة، كما تسمح بتغيير قيمة المتغير واختبار كيف سيكون سلوك بقية البرنامج بناء على هذا التغيير.

كما يمكن للمبرمج تعديل التعليمات في النافذة الوسيطة بنفس الطريقة التي يستخدمها في نافذة تحرير التعليمات Code Editor. ولتقييم التعبيرات أو المتغيرات، يمكن للمستخدم أن يقوم بكتابة علامة استفهام (؟) متبوعة بالمتغير أو بالتعبير المطلوب تقييمه، ثم يضرب المفتاح Enter من لوحة المفاتيح لتظهر النتيجة في السطر اللاحق.

في التدريب التالي نتعرف كيف يمكن اختبار التعليمات باستخدام النافذة الوسيطة.

1.41.1تدريب: اختبار التعليمات في النافذة الوسيطة

229. من القائمة File أنقر فوق New Project، من نافذة New Project أختار Template ومنها Windows Application ثم أنقر المفتاح Ok.

230. من صندوق الأدوات قم بسحب عنصر تحكم من النوع TextBox وعنصر تحكم من النوع Button ثم أرميهم على النموذج.

231. أنقر مرتين فوق المفتاح لتفتح محرر التعليمات على معالج الحدث Button_Click.

232. أضف التعليمات التالية:

```
Dim miles As Integer = 0
Dim hours As Integer = 0
Dim speed As Integer = 0
miles = CInt(Textbox1.Text)
hours = CInt(Textbox2.Text)
speed = miles / hours
MsgBox(CStr(speed) & " miles per hour")
```

أضرب المفتاح F5 لتنفيذ البرنامج. أدخل القيمة ١٠٠ في صندوق النص الأول والقيمة ٠ في صندوق النص الثاني.

أنقر المفتاح Button1 يتوقف البرنامج عن العمل وتظهر نافذة مساعد الاستثناء Exception Assistant مع رسالة خطأ من النوع " Overflow Exception was unhandled "

في النافذة الوسيطة تحت بيئة التطوير المتكاملة IDE – أنظر شكل 6-51 - اكتب ؟ miles ثم أضرب المفتاح Enter، تظهر القيمة ١٠٠ في السطر اللاحق.

أكتب ؟hours ثم أضرب المفتاح Enter. تظهر القيمة ٠ في السطر اللاحق.

أكتب hours=4 ثم أضرب المفتاح Enter، ثم اكتب ؟hours وضرب Enter ثانية. تلاحظ أن قيمة hours أصبحت ٤. وبهذه الطريقة قد قمنا بتغيير قيمة hours بدون أن نغير البرنامج الأصلي.

أضرب المفتاح F5 ليتم استكمال البرنامج وتظهر النتيجة في الرسالة.



شكل 6-51: النافذة الوسيطة

1.42كشف الأخطاء المنطقية

يتوفر في Visual Basic أداتين لاستكشاف الأخطاء هما الـ Breakpoints أو نقاط الإيقاف، والتشغيل خطوة بخطوة stepping.

يمكن للمبرمج أن يقوم بإضافة نقطة توقف Breakpoint عند أي سطر من التعليمات، وعند تنفيذ البرنامج، يتوقف التنفيذ عند نقطة التوقف، وينتقل العمل إلى بيئة التطوير المتكاملة IDE في وضعية الإيقاف Break Mode، حيث يمكن للمبرمج التعرف على أي معلومات خاصة بالبرنامج لحظة توقفه. فيمكن التعرف على قيمة متغير أو اختبار تعبير في النافذة الوسيطة كما تقدم، أو تعديل التعليمات باستخدام خاصية Edit and Continue.

بمجرد انتقال البرنامج إلى وضعية إيقاف يمكن للمبرمج تنفيذه خطوة بخطوة ليتم تنفيذ سطر من التعليمات ثم السطر التالي وهكذا، وذلك بضرب المفتاح F7 في لوحة المفاتيح ليتم تنفيذ السطر ثم ضرب نفس المفتاح مرة أخرى لتنفيذ السطر التالي وهكذا.

إذا كان السطر يحتوي على وظيفة Function أو إجراء فرعي Sub Procedure، فإنه عند النقر على F8 يتم الانتقال إلى داخل هذه الوظيفة أو الإجراء الفرعي. وعند انتهاء تعليمات الوظيفة أو الإجراء الفرعي يعود التنفيذ إلى البرنامج الرئيسي. كما يمكن للمبرمج أن يتفادى الدخول إلى داخل الإجراء وذلك عن طريق أن يضرب المفتاحين SHIFT+F8 معاً.

1.42.1 تدريب: اكتشاف خطأ منطقي

233. من القائمة File أنقر فوق New Project، من نافذة New Project أختار Template ومنها Windows Application ثم أنقر المفتاح Ok.

234. من صندوق الأدوات قم بسحب عنصر تحكم من النوع TextBox وعنصر تحكم من النوع Button ثم أرميهم على النموذج.

235. أنقر مرتين فوق المفتاح لتفتح محرر التعليمات على معالج الحدث Button_Click.

236. أضف التعليمات التالية:

```
Dim minutes As Integer = CInt(Textbox1.Text)
Dim miles As Double = CDbl(Textbox2.Text)
Dim hours As Double = 0
hours = minutes / 60
```

```
MsgBox("Average speed " & GetMPH(hours, miles))
```

أضف تعليمات لعمل هذه الوظيفة:

```
Function GetMPH(ByVal miles As Double, ByVal hours As
Double) _
As String
   GetMPH = CStr(miles / hours)
End Function
```

أضرب المفتاح F5 لتنفيذ البرنامج، أكتب ١٠ في عنصر التحكم TextBox1 و ٥ في عنصر التحكم TextBox2، ثم اضرب المفتاح Button1. لتظهر الرسالة "Average speed 0.03333334" بالرغم من الإجابة الصحيحة هي ٣٠ ميل في الساعة. وهذا يدل على وقوع خطأ في منطق البرنامج.

1.42.2تدريب: إضافة نقاط الإيقاف إلى تعليمات البرنامج

أنتقل إلى محرر التعليمات، ،ابحث عن السطر الذي يحتوي التعبير = hours minutes / 60 وأنقر بمؤشر الفأرة فوقه ثم اضرب المفتاح F9 لإضافة نقطة إيقاف. السطر يتم تظليله باللون الأحمر القاتم علامة على وجود نقطة إيقاف.

أضرب المفتاح F5 لتنفيذ البرنامج وأكتب ١٠ في صندوق النص الأول وخمسة في الصندوق الثاني، ثم أنقر المفتاح Button1. البرنامج يتوقف عند بلوغ نقطة الإيقاف ويصبح السطر الذي يحتوي عليها مظللاً باللون الأصفر. قف بمؤشر الفأرة فوق المتغير hours تظهر لك ملحوظة tip بأن قيمة هذا المتغير هي ٠، كرر نفس الشئ بالنسبة للمتغير minutes لتحصل على القيمة ١٠.

أضرب المفتاح F8 لتنفيذ السطر الذي يحتوي على التعبير = hours minutes / 60والانتقال إلى السطر الذي يليه وتحقق بنفس الطريقة السابقة من قيم المتغيرات في العبارة MsgBox("Average speed " & GetMPH(hours, miles)) حيث من المفترض أن تكون قيمة hours هي ٠.١٦٦٦٦٦٦٧٢ وقيمة miles هي ٥.٠.

أضرب المفتاح F8 لتنفيذ السطر التالي، لاحظ أن التنفيذ أنتقل إلى داخل الوظيفة GetMPH، قم بالتعرف على قيمة المتغيرات تلاحظ أن المتغير miles أصبحت قيمته ٠.١٦٦٦٦٦٦٧٢ بينما أصبحت قيمة المتغير hours هي 5.0 أي أنه حدث خطأ في تخصيص قيم المدخلات وهكذا تولد الخطأ المنطقي.

1.42.3 تدريب: علاج الخطأ المنطقي

قم بتعديل هذا السطر ليصبح في الصورة

```
MsgBox("Average speed " & GetMPH(hours, miles))
```

قم بإزالة نقطة الإيقاف وذلك بأن تنقل مؤشر الفأرة للسطر الذي يحتوي على نقطة الإيقاف وأضرب المفتاح F9.

أعد تشغيل البرنامج وأختبر نتائجه.

1.43 التعليقات Comments

التعليقات هي سطور يضيفها المبرمج ليبين لنفسه أو لشركائه في تطوير التطبيق ما الذي يجب أن تختزنه المتغيرات من بيانات والعمليات التي تنفذها تعليمات البرنامج المختلفة. وأهمية التعليقات أنها تجعل البرنامج مفهوم بالنسبة للمطور أو المطورين الآخرين مما يسهل مهامهم التطويرية. ويعتبر أي سطر يبدأ برمز الفاصلة العالية ' هو بمثابة تعليق لا يلتفت إليه البرنامج عند التنفيذ بينما يقوم محرر التعليمات بعرضه باللون الأخضر الفاتح دلالة على أن هذا السطر من التعليقات.

مقدمة إلى تطبيقات قواعد البيانات

الغرض من إنشاء أي تطبيق هو معالجة البيانات. في معظم التطبيقات التي استخدمناها كأمثلة في الفصول السابقة كانت البيانات تدخل يدوياً ثم تختزن في متغيرات داخل البرنامج. لكن في كثير من التطبيقات الحقيقية يتم اختزان البيانات في خارج التطبيق في أشكال مختلفة مصممة لاختزان البيانات. و ما أهم هذه الأشكال قواعد البيانات Database.

في هذا القسم سوف نتعرض إلى تطوير تطبيقات قواعد البيانات باستخدام Visual Basic.NET 2008 مع مقدمة بسيطة إلى تقنية Linq التي تقدمها Microsoft لأول مرة في Visual Studio 2008.

1.44 قواعد البيانات Database

هذا القسم ليس الغرض منه تقديم قواعد البيانات بصورة تفصيلية، ولكن تقديم مفاهيمها الأساسية حتى يمكن للدارس أن يفهم ما هو مطلوب منه بنهاية هذا القسم.

تعرف قاعدة البيانات بأنها تجميع لبيانات منظمة لتمثيل ظاهرة ما. تتكون قاعدة البيانات من جداول Tables يمثل كل جدول منها جزء من الظاهرة، ويتكون كل جدول من حقول Fields (أو أعمدة Columns) و سجلات Records (أو صفوف)، حيث تمثل السجلات أوصاف العناصر المختلفة التي يصفها الجدول، بينما تمثل الحقل وصف محدد لجميع الحالات. وترتبط هذه الجداول بعلاقات بينها تماثل العلاقات في العالم الحقيقي.

وسنضرب مثالاً لتقريب أفكار قواعد البيانات للذهن، لنفترض أن هناك قاعدة بيانات الغرض منها وصف مدرسة، إذا فالمدرسة هي الظاهرة المطلوب تمثيلها. هناك أربعة موضوعات أساسية عند وصف المدرسة هي المدرسون والطلاب والفصول والمواد التعليمية. كل موضوع من هذه المواضيع سيتم حفظه في جدول مستقل. فمثلاً جدول المدرسون سيحتوي على أعمدة عن أسم المدرس ودرجته الوظيفية والمادة العلمية التي يقدمها. بينما كل صف سيمثل سجل المدرس. كما أن هناك جدول سيمثل الطلاب. وآخر

للفصول، ولأن كل فصل ينشغل بعدد من الطلاب، فإن هناك رابطة يجب إنشاءها لتبين مكان الطالب في أي فصل.

1.45أنشاء قاعدة بيانات

غالباً ما يتم أنشاء قواعد البيانات باستخدام برامج خاصة مثل Access أو SQL Server وبصورة مستقلة تماماً عن Visual Studio ثم يتم كتابة التطبيقات التي تعمل بمثابة واجهات استخدام لقواعد البيانات باستخدام Visual Basic. وحيث أذنا لسدنا في معرض الحديث عن برامج إدارة قواعد البيانات فإن Visual Studio.NET 2008 يتوفر معه نسخه مخففة لبرنامج إدارة قاعدة البيانات SQL Server Compact Version لاستخدامها في إنشاء قواعد البيانات. و هو البرنامج الذي سوف نقوم باستخدامه في هذا القسم.

1.45.1أنشاء قاعدة البيانات

237.من File أختر New Project.

من نافذة New Project أختار Windows Form Application.

الاسم التلقائي في الخانة Name هو WindowsApplication قم بتغييره إلى DBApp.

أنقر المفتاح OK.

من القائمة Project أختر Add New Item لتظهر النافذة المبينة في شكل 7-52.

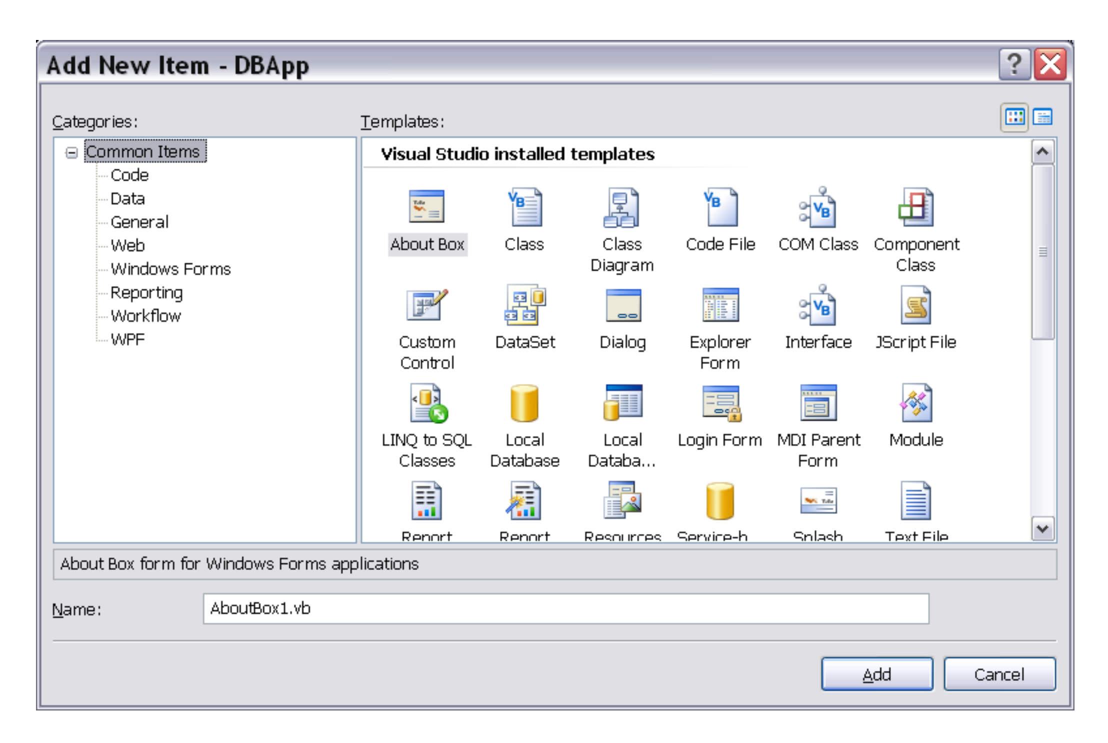
أنقر فوق Local Database.

في الخانة Name أكتب اسم قاعدة البيانات وهو FirstDB.

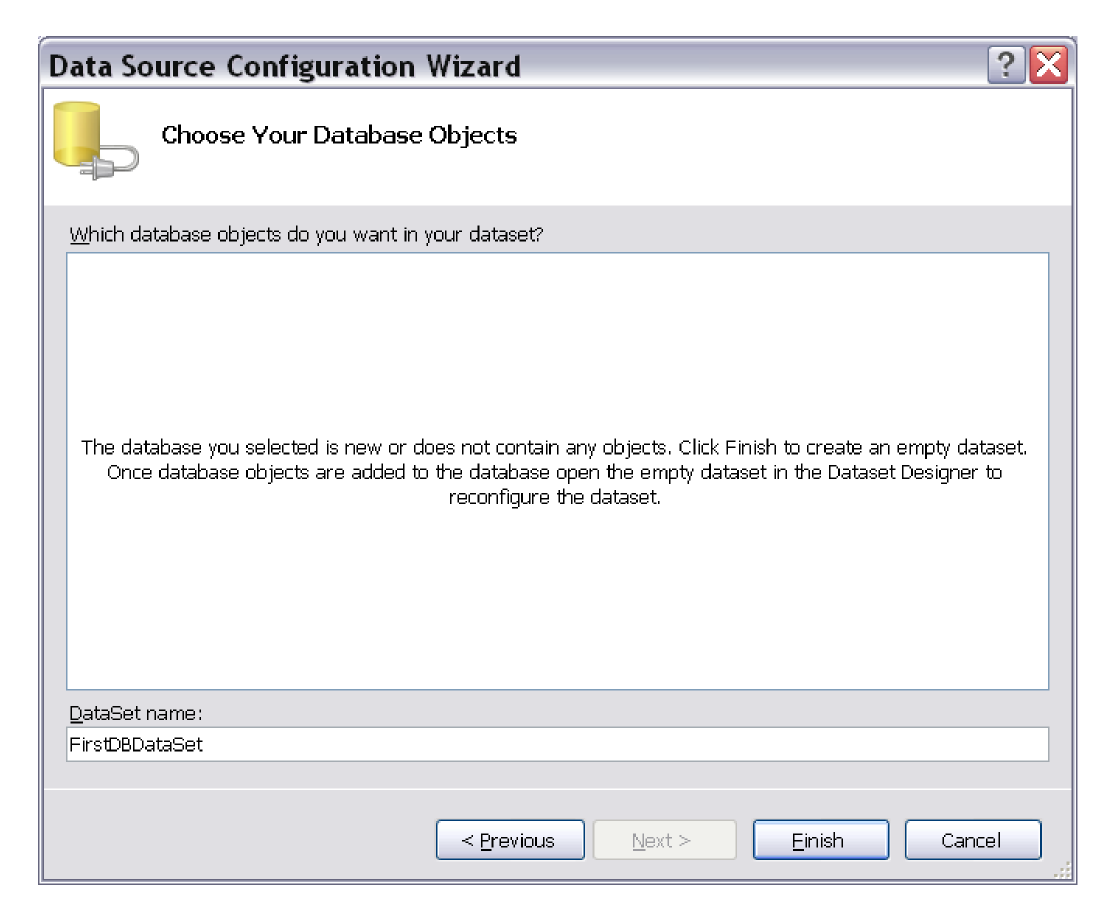
أنقر Add.

تظهر نافذة معالج Data Source Configuration Wizard المبذية في شكل 7-53 أختر Cancel.

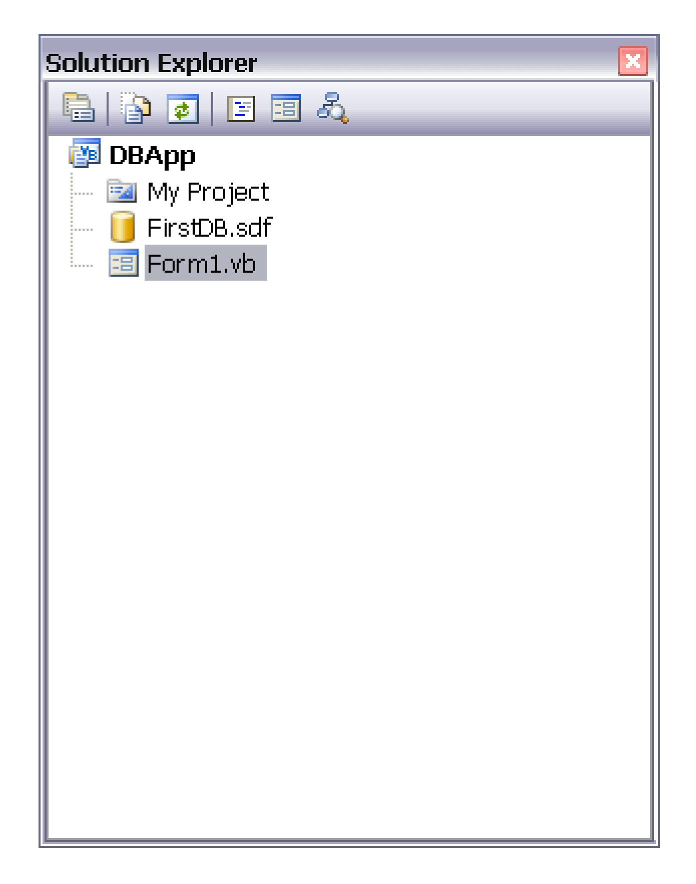
تظهر قاعدة البيانات الجديدة FirstDB في نافذة Explorer Solution كما في شكل 7-54.



شكل 7-52: نافذة Add New Item



شكل 7-53: نافذة Data Source Configuration Wizard



شكل 7-54: قاعدة البيانات FirstDB في نافذة Solution Explorer.

1.45.2إضافة جدول إلى قاعدة البيانات

من القائمة View أختار Server Explorer. لتظهر هذه النافذة كما هي مبينة في شكل 7-55.

أنقر فوق علامة + المجاورة لقاعدة البيانات FirstDB.

أنقر يمين فوق Tables واختار Create Table. لتظهر نافذة New Table المبينة في شكل 7-56.

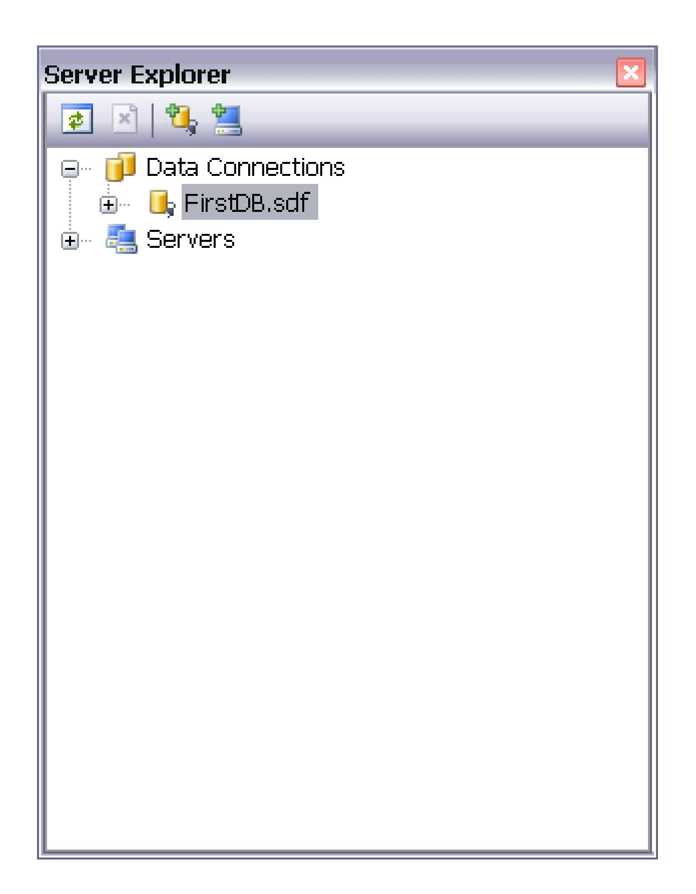
في الخانة Name حدد أسم الجدول وليكن Addresses.

أنقر تحت Column Name واكتب أسم الحقل الأول وهو FirstName.

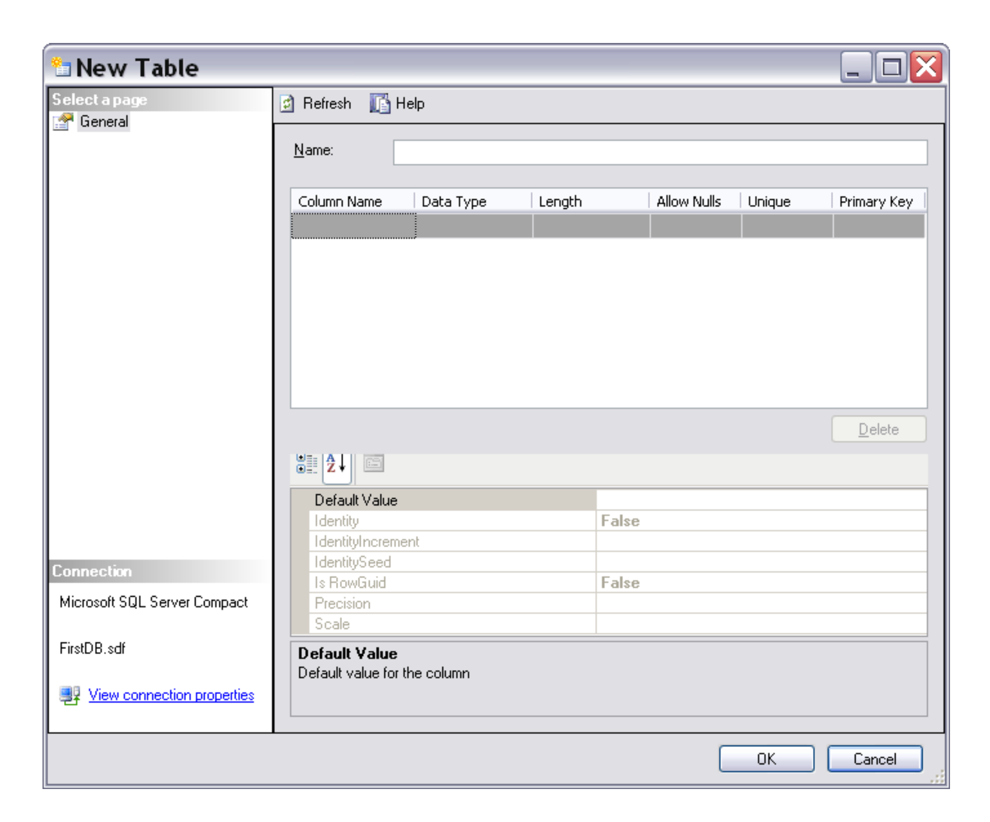
دع الـ Data Type كما هو.

غير قيمة الـ Length لتصبح 50.

قم بإنشاء حقول لتكون خصائصها كما هي بينة في الجدول التالي:



شكل 7-55: نافذة Server Explorer



شكل 7-56: نافذة New Table.

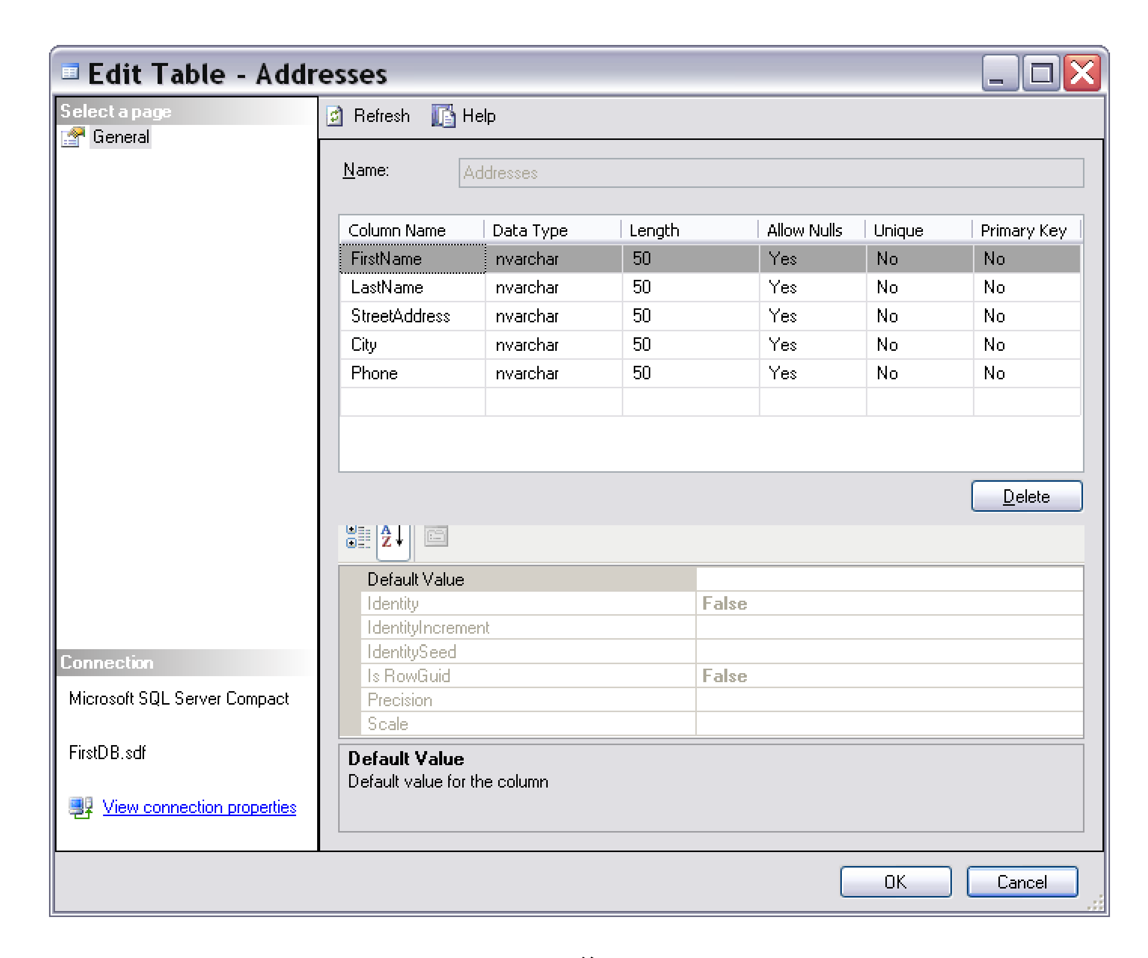
اسم الحقل	نوع البيانات	طول الحقل
LastName	nvarchar	50
StreetAddress	nvarchar	50
City	nvarchar	50
Phone	nvarchar	50

أنقر المفتاح OK ليتم إنشاء الجدول.

1.45.3إضافة المفتاح الأساسي Primary Key

في النافذة Explorer Server أختار Tables ثم من ها اختار الجدول Address.

أذقر يمين وأختار Edit Table Scheme لتظهر النافذة المبينة في شكل 7-57.



شكل 7-57: نافذة Edit Table

أمام اسم الحقل FirstName وتحت الحقل Allow NullS غير القيمة من Yes إلى No، وغير قيمة Primary Key لتصبح Yes.

كرر ذات الإجراء السابق على الحقل LastName.

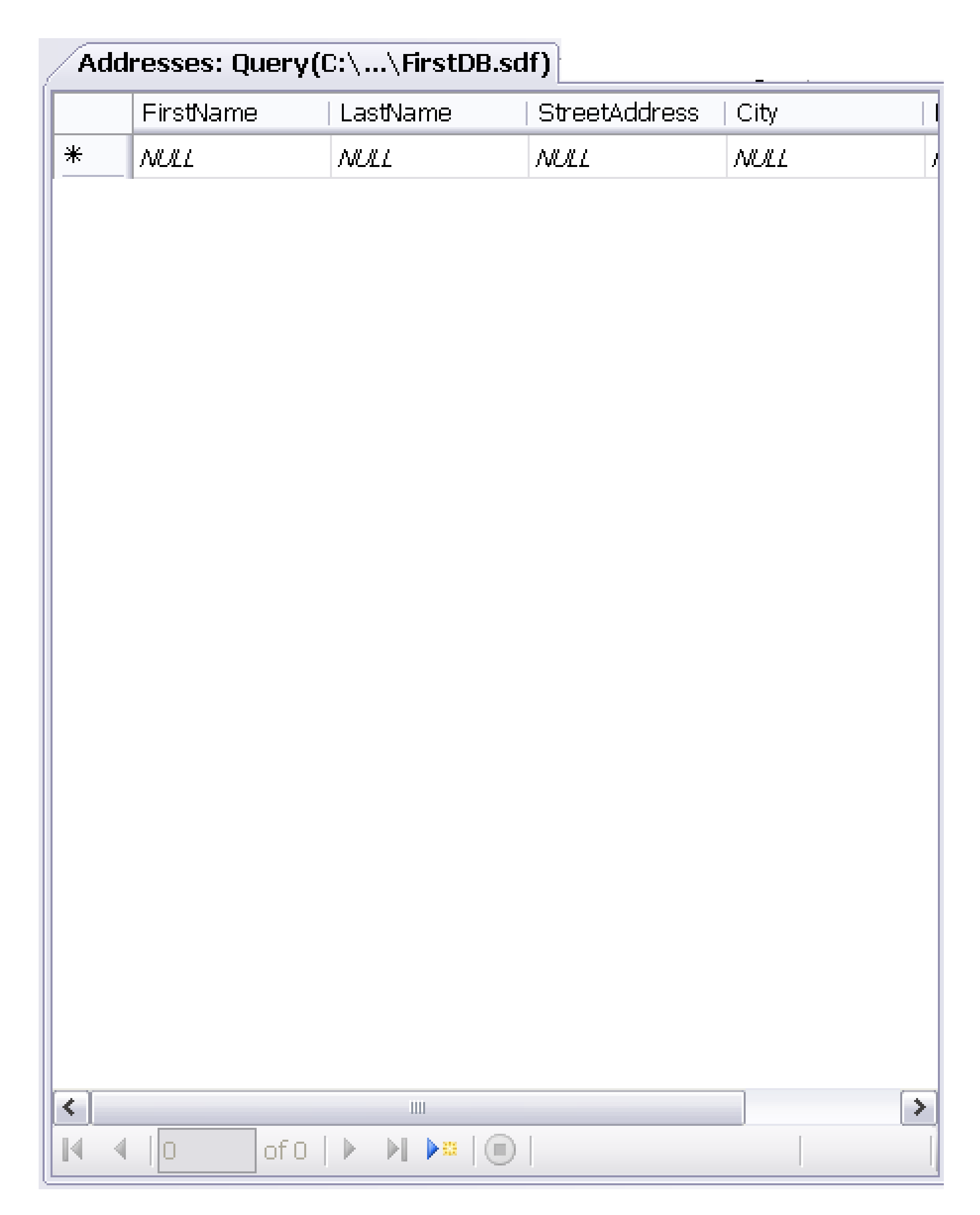
أنقر المفتاح OK.

بهذه الطريقة جعلنا من الحقلين FisrtName و LastName حقلي مفتاح أساسي.

1.45.4إضافة البيانات إلى الجدول

في النافذة Server Explorer أختار Tables ثم منها اختار الجدول Address.

أنقر يمين وأختار Show Table Data لتظهر النافذة المبينة في شكل 7-58.



شكل 7-58: نافذة عرض وإدخال البيانات

قم بإدخال البيانات حتى تكون مثل المبينة في شكل 7-59.

	e	LastName	StreetAddress	City	Phone
		Ali	25 El Horeya ...	Alexandria	5085080
		Karim	59 Canal Str	Zagazig	8989009
		Mohammed	171 Zohour St	Alexandria	4545454
		Khalil	23 Ibn Maja St.	El Mansoura	7894664
		Gharib	25Mouhata St	Alexandria	7896541
▸*		NULL	NULL	NULL	NULL

شكل 7-59: البيانات المدخلة للجدول

أخيراً من قائمة File أختار Save All.

1.46الاتصال بقاعدة البيانات

في هذا القسم نتعلم كيف يمكن للتطبيق أن يتصل بقاعدة البيانات، و من ثم استخدام أدوات Visual Studio.NET لتصفح محتويات قاعدة البيانات وإضافة نسخة من محتوياتها للتطبيق نفسه، و هو ما سنقوم به في التدريبات التالية حيث سوف نقوم بإنشاء تطبيق يتصل بقاعدة البيانات ألتي أنشلناها من قبل.

تأكد أن قد قمت بإغلاق التطبيق السابق قبل تنفيذ الإجراءات التالية.

238.من القائمة File أختار New Project.

أختار Windows Form Application.

في الخانة Name أكتب اسم التطبيق AddressApp.

أنقر المفتاح OK.

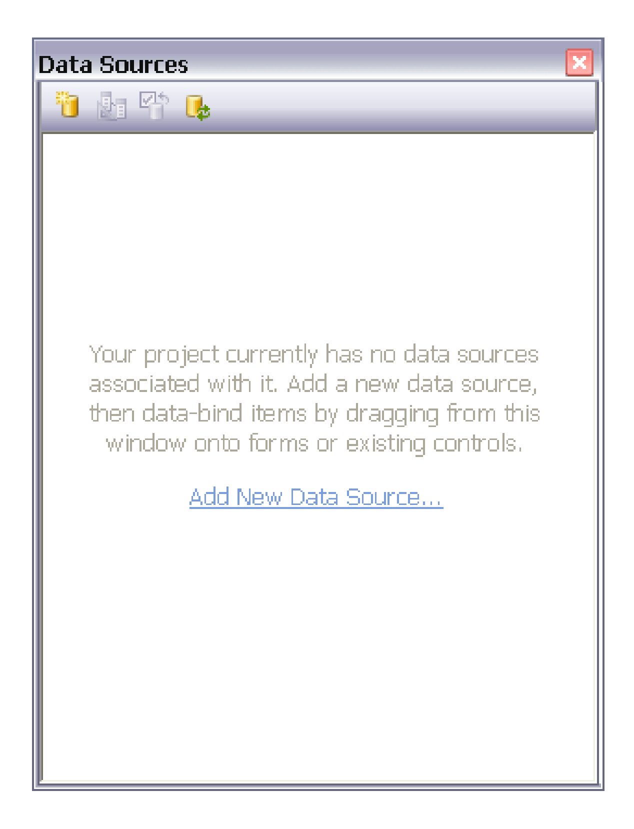
من القائمة Data أختار Show Data Sources. تظهر النافذة كما هي مبينة في شكل 7-60.

في نافذة Data Sources أنقر Add New Data Source. يظهر المعالج Data Source Configuration Wizard المبينة في شكل 7-61.

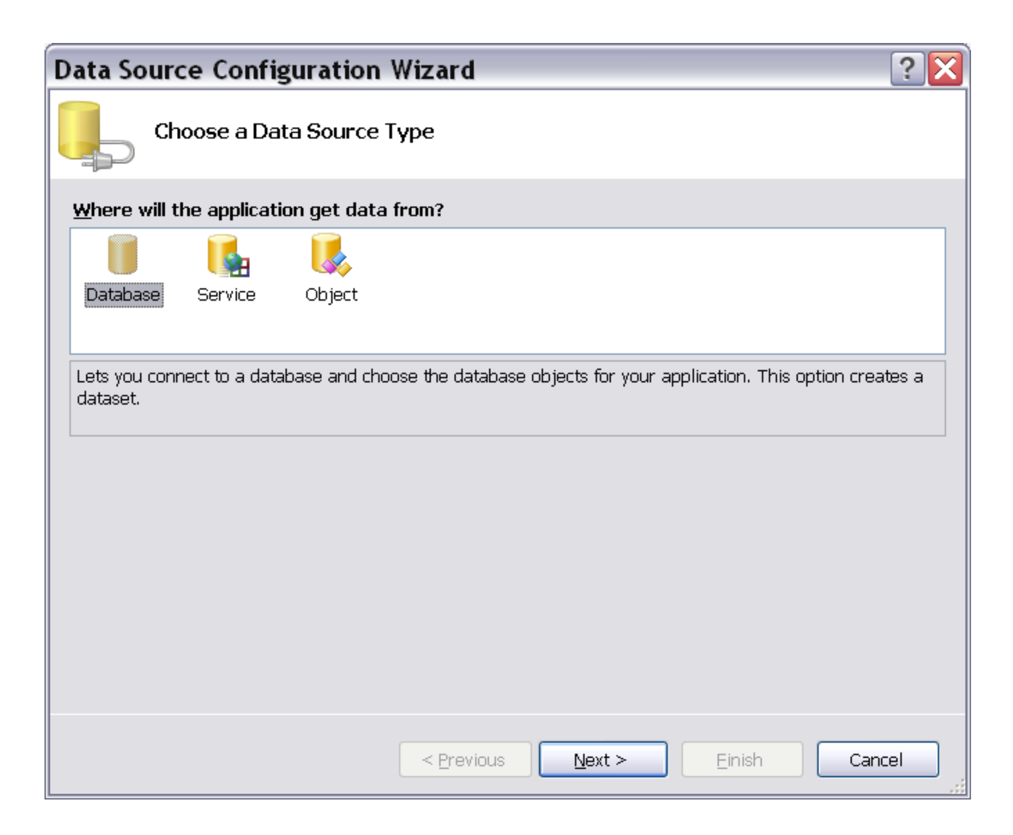
أختار Database وأنقر Next.

أنقر المفتاح New Connection. تظهر نافذة Choose Data Source المبينة في شكل 7-62.

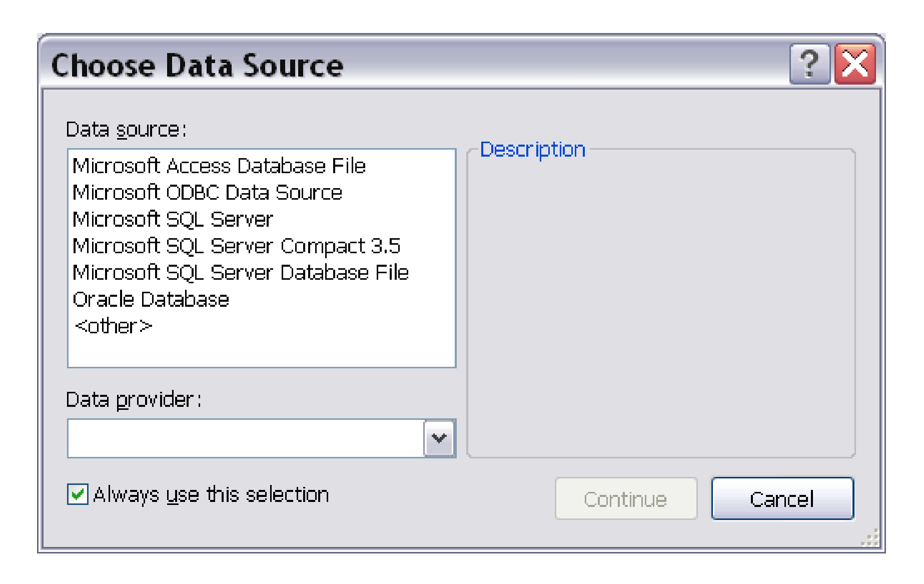
من القائمة Data Source أختار Microsoft SQL Server Compact 3.5.



شكل 7-60: نافذة Data Sources



شكل 7-61: نافذة المعالج Data Source Configuration.



شكل 7-62: النافذة Choose Data Source.

أنقر المفتاح Continue. تختفي النافذة Choose Data Source وتظهر النافذة Add Connection المبينة في شكل 7-63.

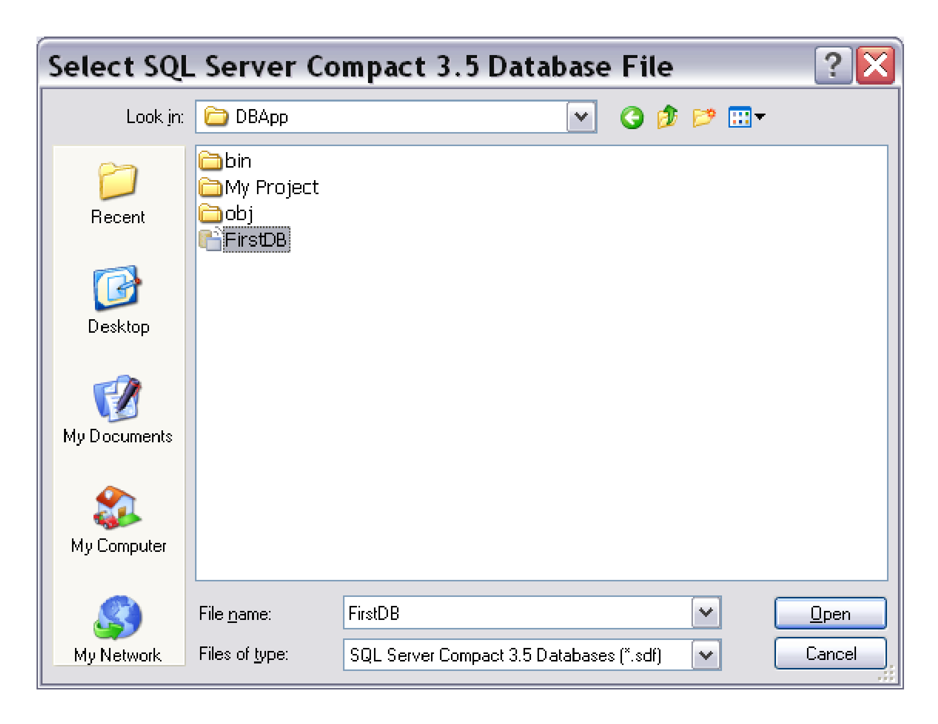
Add Connection
Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.
Data source:
Microsoft SQL Server Compact 3.5 (.NET Fra
Change...
Data Source
My computer
ActiveSync connected device
Connection Properties
Database:
Create...
Browse...
Password:
Save my password
Advanced...
Test Connection
OK
Cancel

شكل 7-63: النافذة Add Connection

أنقر المفتاح Browse.

في نافذة اختيار الملف أنتقل للمكان الذي حفظ عليه التطبيق الذي سبق أن قمنا بإنشائه في 1.45 ثم أختار قاعدة البيانات FirstDB – أنظر شكل 7-64.

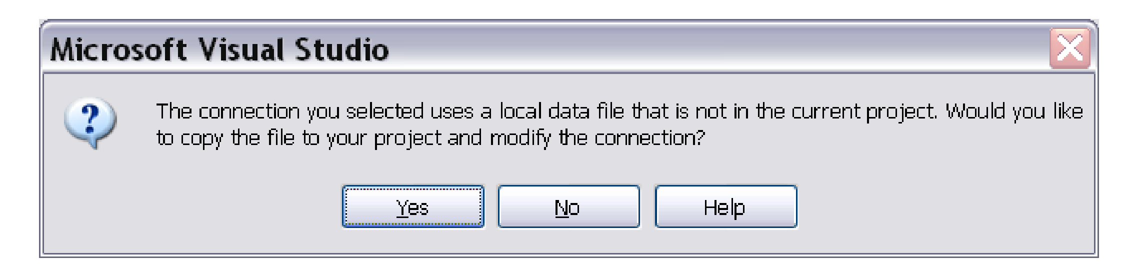
أنقر المفتاح Open.



شكل 7-64: اختيار ملف قاعدة البيانات.

أنقر المفتاح Ok حتى يتم إغلاق النافذة Add Connection ونرجع لنافذة معالج Data Source Configuration Wizard.

أنقر Next لتظهر الرسالة المبينة في شكل 7-65.



شكل 7-65: رسالة عن إمكانية نقل البيانات إلى المشروع.

تسأل هذه الرسالة عما إذا كنت ترغب في نقل البيانات إلى المشروع، أختار Yes.

في النافذة التالية من المعالج Data Source Configuration Wizard والمبينة في شكل 7-66 تأكد من اختيار Yes, save connection as ودع الاسم المقترح كما هو وانقر Next.



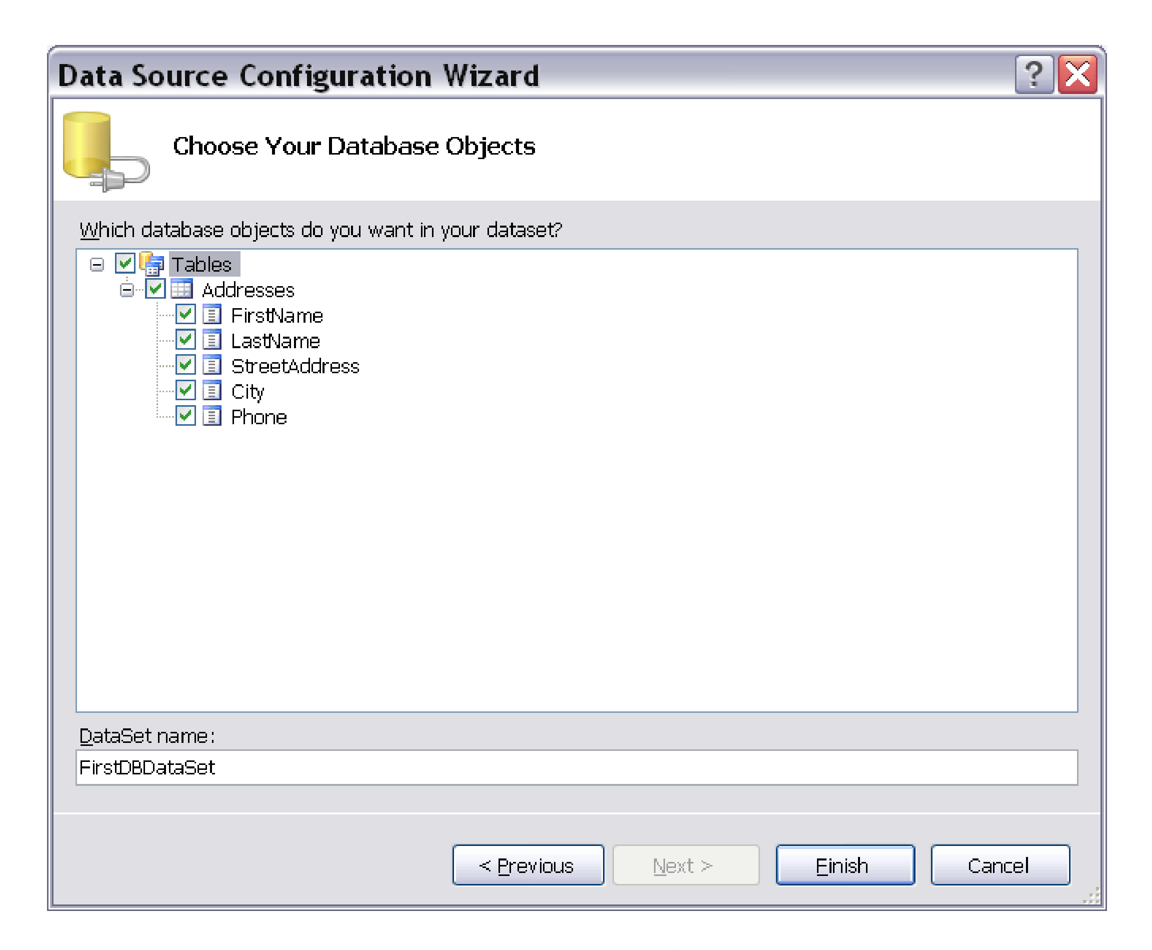
شكل 7-66: حفظ الوصلة

في الصفحة Choose your database objects من المعالج Data Source Configuration Wizard – أنظر – قم بالتأشير على جميع محتويات جدولك.

أنقر Finish.

تظهر قاعدة البيانات FisrtDB في الـ Solution Explorer، كما تظهر الوصلة FirstDBDataSource في نافذة Data Sources.

من القائمة File أختار Save All.



شكل 7-67: الصفحة Choose Your Database Objects

1.47 عرض البيانات في واجهة رسومية

في القسم السابق تعلمنا أن نقوم بعمل وصلة لقاعدة البيانات، في هذا القسم نتعلم كيف نستخدم هذه الوصلة حتى نتمكن من بلوغ البيانات وعرضها في واجهة رسومية بالتطبيق خاصتنا.

239. أفتح التطبيق السابق المسمى AddressApp.

من النافذة Solution Explorer أختار Form1.vb.

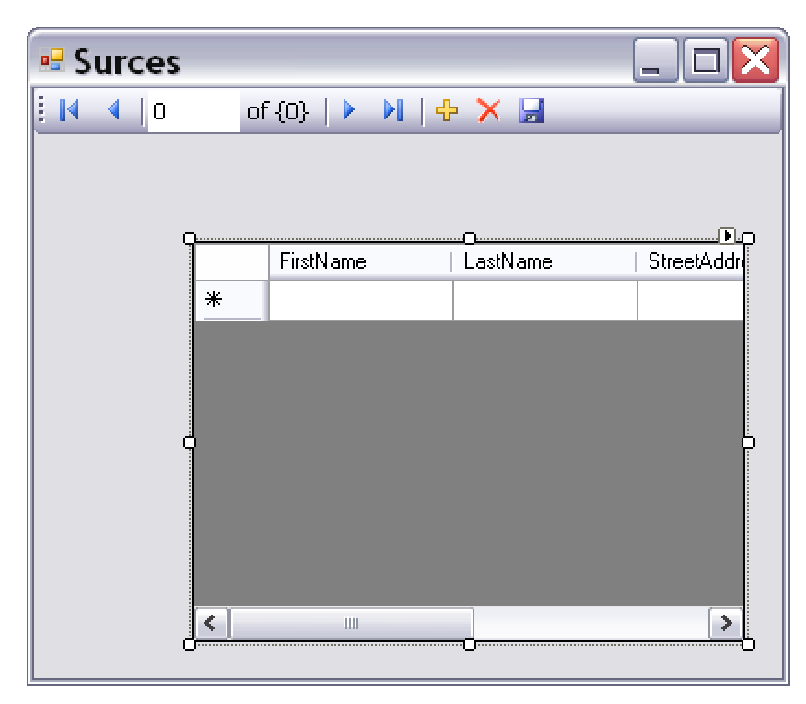
أنقر فوق View Designer.

أنقر فوق Data Source.

من النافذة Data Source قم بسحب الجدول Address ورميه فوق النموذج Form1.



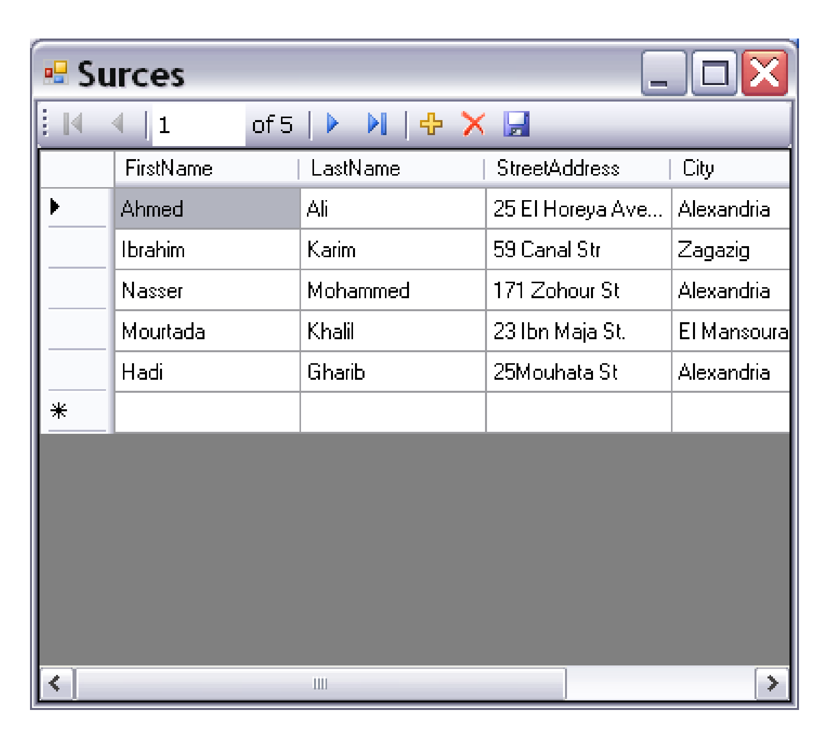
تظهر نتيجة لذلك عدد من عناصر التحكم كما هو مبين في شكل 68-7.



شكل 7-68: عناصر التحكم التي أضيفت للنموذج Form1.

أختار عنصر التحكم AddressesDataGridView و من نافذة الخصائص أجعل الخاصية Dock تساوي Fill.

أضرب المفتاح F5 في لوحة المفاتيح لتنفيذ البرنامج.



شكل 7-69: البرنامج عند التشغيل.

1.48تحديث البيانات

في هذا القسم سوف نتعلم كيف يمكننا أنشاء واجهة لإدخال وتحديث البيانات في قاعدة بيانات محلية. فيما سبق قمنا بإنشاء نسخة من قاعدة البيانات في التطبيق وأطلقنا عليها أسم قاعدة البيانات المحلية، وهذا يعني أن هذه البيانات ليست هي تلك المحفوظة في قاعدة البيانات الخارجية ولكنها مجرد صورة منسوخة عنها محفوظة في المشروع يطلق عليها اسم Dataset. و في كل مرة نقوم فيها بتشغيل البرنامج يتم عمل صورة عن قاعدة البيانات الأصلية وتصبح هي الـ Dataset لكن عندما يتم تعديل الـ Dataset فإن هذا لا ينعكس في قاعدة البيانات الخارجية.

لعلك لاحظت في واجهة التطبيق المبينة في شكل 7-69 أن هناك أيقونة Save، عندما يقوم المستخدم بالنقر فوق Save فإن البيانات المدخلة إلى الـ Dataset يمكن حفظها في قاعدة البيانات الخارجية. لكن ماذا إذا نسى المستخدمين النقر فوق هذا المفتاح؟!. لتجنب مثل هذه المواقف يمكن للمبرمج أن يقوم بإضافة تعليمات لحفظ البيانات عند إغلاق البيانات وهذا ما سوف نقوم به هنا.

240.أفتح المشروع السابق.

من نافذة Solution Explorer أختار قاعدة البيانات FirstDB وتأكد من خصائصها أن الخاصية Copy to Output Directory فعالة ومضبوطة على Copy if Newer.

أختار النموذج Form1.

أختر View Designer.

أختر AddressesDataGridView من فوق النموذج ثم أضرب المفتاح Delete من لوحة المفاتيح لحذفه.

أختار التبويب Data Source في Solution Explorer.

أنقر فوق علامة + إلى جوار قاعدة البيانات FirstDB لتظهر محتوياتها.

قم بس حب ور مي كل عنصر من عنا صر قاعدة البيانات فوق النموذج Form1، يظهر كل مرة عنصري تحكم واحد من النوع Label والآخر من النوع TextBox أنظر شكل 7-70.

شكل 7-70: تعديل واجهة التطبيق.

أنقر مرتين فوق النموذج لتفتح محرر التعليمات Code Editor.

أختار من قائمة الأحداث Events الحدث FormClosing.

في معالج الحدث Form1_FormClosing اكتب التعليمات التالية:

```
Me.AddressesBindingSource.EndEdit()
Me.AddressesTableAdapter.Update(Me.FirstDBDataSet.Addre
sses)
```

هذه التعليمات تجعل الكائن AddressesTableAdapter ينسخ أي تعديلات البيانات تحدث في الـــ Dataset إلى قاعدة البيانات الأساسية.

أضرب المفتاح F5 لتنفيذ البرنامج. قم بتغيير بعض البيانات وأضف بعض السجلات ثم أغلق النموذج.

أضرب المفتاح F5 لتنفيذ البرنامج وتأكد أن التغييرات التي قمت بها قد نفذت.

1.49عرض البيانات من جداول مترابطة

في الأقسام السابقة تعلمنا أن نقوم بإنشاء نماذج لإدخال وتحديث البيانات إلى قاعدة البيانات. في هذا القسم نتناول كيف يمكن عرض البيانات من جدولين منفصلين في نموذج.

1.49.1تدريب: الاتصال بقاعدة البيانات Northwind.

241.من القائمة File أختار New Project.

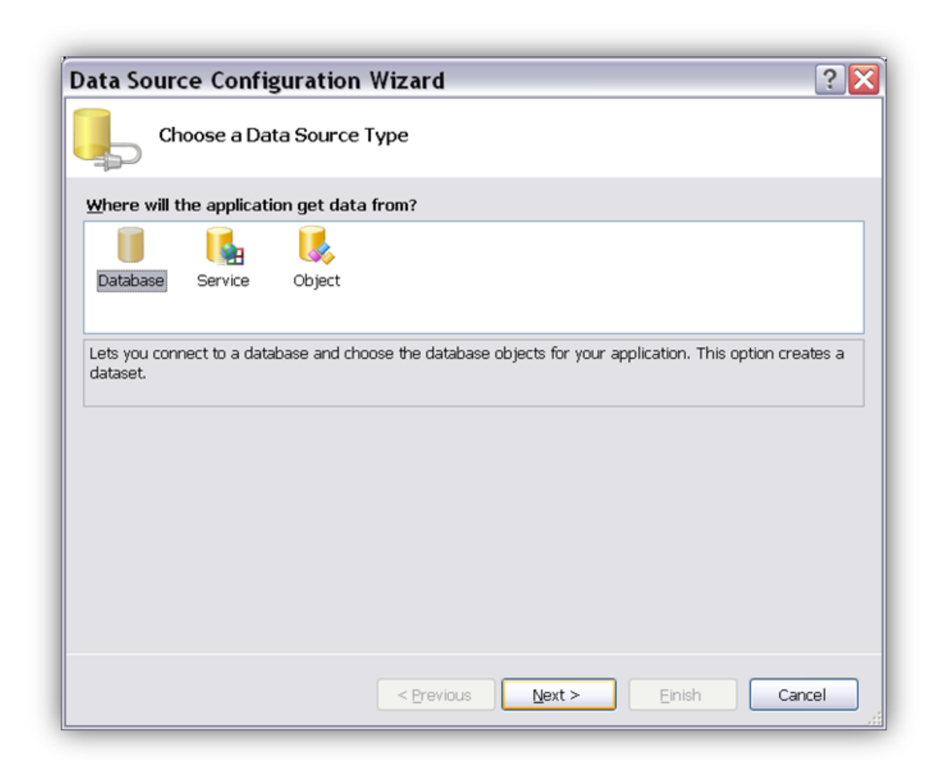
أختار Windows Form Application.

أنقر المفتاح OK.

أختار من التبويب Data Sources في نافذة Solution Explorer.

أنقر فوق Add New Data Source.

تظهر نافذة Data Source Configuration Wizard المبينة في شكل 7-71



شكل 7-71: نافذة Data Source Configuration Wizard

أختار Database.

أنقر Next.

أنقر فوق المفتاح New Connection.

تظهر النافذة Add Connection المبينة في شكل 7-72



شكل 7-72: النافذة Add Connection

أنقر فوق المفتاح Browse.

من المجلد Program Files\Microsoft SQL Server Compact\Edition\v3.5\Samples\ أختار قاعدة البيانات Northwind.

أنقر المفتاح Open.

في النافذة Add Connection أنقر OK.

في النافذة نافذة Data Source Configuration Wizard أنقر Next.

أجب بـ Yes عن السؤال الذي سوف يظهر.

تأكد أن المربع المعنون Yes, save the connection as معلم.

في الصفحة التالية والمعنونة بـ Choose Your Database Objects أنقر فوق علامة + بجوار Tables لترى جميع الجداول، ثم أنقر المربعين المجاورين لـ Order Details و Products.

أنقر المفتاح Finish.

يتم إضافة قاعدة بيانات محلية للمشروع هي NorthwindDataSet وتظهر في التبويب Data Sources.

أنقر فوق NorthwindDataset في التبويب Data Sources.

في النافذة Properties أجعل الخاصية Copy to Output Directory لها القيمة Copy if newer.

في القائمة File أختر Save All وأحفظ المشروع.

1.49.2تدريب: عرض البيانات المترابطة

يمكن من خلال التبويب Data Sources استكشاف العلاقة Relationship بين الجدولين المترابطين. في هذا التدريب مثلاً إذا نقرت فوق الجدول Products ستجد أن الجدول Order Details موجود في نهاية الجدول Product إشارة إلى أن هناك ثمة علاقة بين الجدول Product الجدول Order Details. وإذا قام المبرمج بسحب الجدول Order Details ورميه فوق النموذج فإن العلاقة بينه وبين الجدول Products تظهر في صورة الاستعراض المشترك للسجلات كما سوف نفعل فيما يلي.

في التبويب Data Sources أنقر فوق علامة + في الجدول Product لاستعراض الحقول التي يحتويها.

إلى جوار الحقل Product Name أنقر على السهم ثم أختار عنصر التحكم Label.

قم بسحب الحقل Product Name من التبويب Data Sources إلى النموذج.

قم بسحب الجدول Order Details من ذيل قائمة محتويات الجدول Product إلى النموذج ليتم عرضه في نافذة جدول.

أنقر F5 لتنفيذ البرنامج.

جرب استعراض محتويات الجدولين.

حول برمجة تطبيقات قواعد البيانات

التقنية المعروضة في هذا القسم تناسب مقدمة مبسطة حول برمجة تطبيقات قواعد البيانات والتي يلزم لتطويرها بصورة مهنية الإلمام بتقنية ADO.NET المكملة لـ Visual Basic.NET 2008 والمتعلقة ببرمجة تطبيقات قواعد البيانات. وهذه التقنية مصاحبة لـ Visual Basic.NET منذ الإصدارة 2003، بينما تميزت Visual Studio.NET 2008 بتقنية جديدة لبرمجة تطبيقات قواعد البيانات هي تقنية Linq.

استخدام الملفات

كما قدمنا من قبل في القسم السابع، تتعامل معظم التطبيقات مع بيانات مختزنة خارج التطبيق، و فق القسم السابع تعرفنا كيف يمكن التعامل مع البيانات المختزنة في قواعد البيانات. لكن هناك كثير من التطبيقات التي تتعامل مع بيانات مختزنة خارجها في أشكال أخرى غير قواعد البيانات أهمها الملفات. في هذا القسم سنتعرف على المهام الأساسية للاستخدام الملفات في Visual Basic.NET 2008، وهذه المهام تعتمد على الكائن My.Computer.FileSystem والذي يسهل على المبرمج التعامل مع الملفات المختلفة[1].

في هذا القسم سوف نقوم ببناء تطبيق يهدف إلى استعراض الصور المختزنة على المجلد My Pictures، كما سنقوم بالتعرف على الكائن My ونتعلم كيف يمكن استخدامه لقراءة وكتابة البيانات في ملف نصي.

1.50عرض الملفات المختزنة في مجلد

في هذا القسم سوف نتعرف على استخدام الكائن My.Computer.FileSystem للتعرف على أسماء الملفات المختزنة في مجلد معين. وبداية سوف نعرف المجلد Folder كمساحة يختزن عليها Microsoft Windows الملفات، ويقدم Microsoft Windows مجموعة من المجلدات المخصصة لحفظ أنواع معينة من الملفات للتبسيط على المستخدم مثل المجلد My Document و My Pictures[2].

وكما سبق أن قدمنا موضوع تدريبنا هو إنشاء متصفح للصور الموجودة في المجلد My Pictures وذلك باستخدام عنصري التحكم List Box لعرض ملفات الصور و Picture Box لعرض الصور نفسها.

---

[1] وبغض النظر عن نظام التشغيل الذي يعمل البرنامج من خلاله.

[2] في Microsoft Windows XP تعرف هذه المجلدات بـ MyDocument و MyPictures وهكذا، بينما في Microsoft Windows Vista تعرف بـ Documents و Pictures وهكذا بدون My، لذلك يجب على القارئ أخذ هذا في الاعتبار عند كتابة التعليمات الخاصة بالتطبيقات الواردة في هذا القسم.

1.50.1تدريب التعرف على الملفات

يمكن أن نستخدم الطريقة My.Computer.FileSystem.FileExists للمعرفة ما إذا كانت هناك ملفات مختزنة في مجلد معين أو لا ، كما يمكن البحث عن ملفات محددة بالاسم بواسطة هذه الطريقة. ولاسترجاع ملف معين يمكن استخدام الطريقة My.Computer.FileSystem.GetFiles لهذا الغرض. كما يمكن استخدام العلامات الخاصة Wild Characters للبحث عن الملفات. كذلك يمكن استخدام الكائن My.Computer.FileSystems.SpecialDirectories لتخصيص مجلد معين مثل MyPictures وهو يتيح للمبرمج أن يسترجع قائمة بأسماء جميع الملفات المختزنة في هذا المجلد.

الآن نفذ الإجراءات التالية لتنفيذ واجهة البرنامج وعرض الملفات المختزنة في My Pictures.

242.من القائمة File أختار New Project.

أختار Windows Form Application.

في الخانة Name أكتب اسم التطبيق PictureViewer.

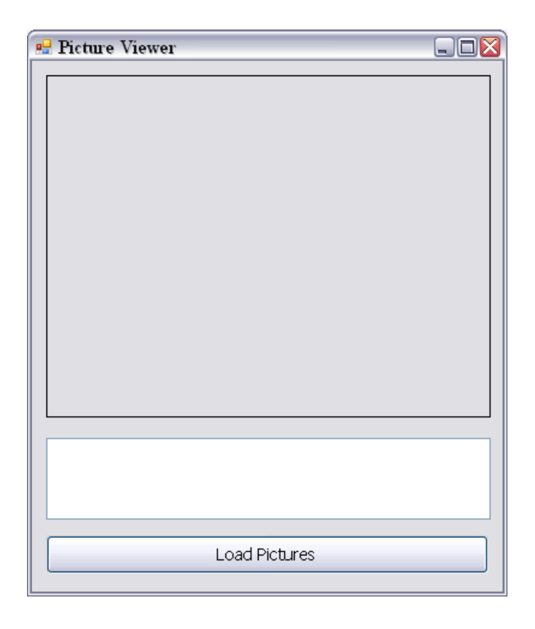
أنقر المفتاح OK.

أضف على النموذج عناصر التحكم ListBox و PictureBox و Button.

أضبط خصائص عناصر التحكم المكونة للواجهة لتتوافق مع الجدول التالي.

العنصر	الخاصية	القيمة
Form1	Text	Picture Viewer
	Size	472 ,400
PictureBox1	BorderStyle	FixedSingle
	SizeMode	StretchImage
	Size	285 ,370
	Location	12 ,12
ListBox1	HorizontalScrollBar	True
	Size	82 ,370
	Location	314 ,12
Button1	Name	LoadPictures
	Text	Load Pictures
	Size	32 ,368

395 ,12	Location	



شكل 8-73: واجهة التطبيق

أنقر فوق المفتاح LoadPictures مرتين لتفتح الـ Event handler الخاص به.

أضف التعليمات التالية إلى الحدث:

```
Me.ListBox1.Items.Clear()
Me.PictureBox1.ImageLocation = ""

For Each founImage As String In _
    My.Computer.FileSystem.GetFiles(_
My.Computer.FileSystem.SpecialDirectories.MyPictures,_
FileIO.SearchOption.SearchTopLevelOnly,_
"*.gif", "*.jpg")
    Me.ListBox1.Items.Add(founImage)
Next
```

في هذه التعليمات تستخدم الطريقة My.Computer.FileSystem.GetFiles لاسترجاع قائمة بأسماء الملفات، وتتعين الملفات المسترجعة بهذه الطريقة بثلاثة مدخلات، الأول هو المجلد المطلوب استرجاع الملفات المختزنة عليه وهو المجلد My.Computer.FileSystem.SpecialDierctories.MyPictures، ثم المتغير

الثا ني طريق ة الب حث كا حد عنا صر العائلة FileIO.SearchOption والعنصر SearchTopLevelOnly يعني البحث في المجلد المحدد دون المجلدات الفرعية، وثالثا نوعية الملفات المطلوب البحث عنها كمتغيرات نصية.

1.50.2عرض الصور

لعرض الصور قم بتنفيذ الإجراءات التالية:

في نافذة Code Editor أختار ListBox1 من القائمة Class Name.

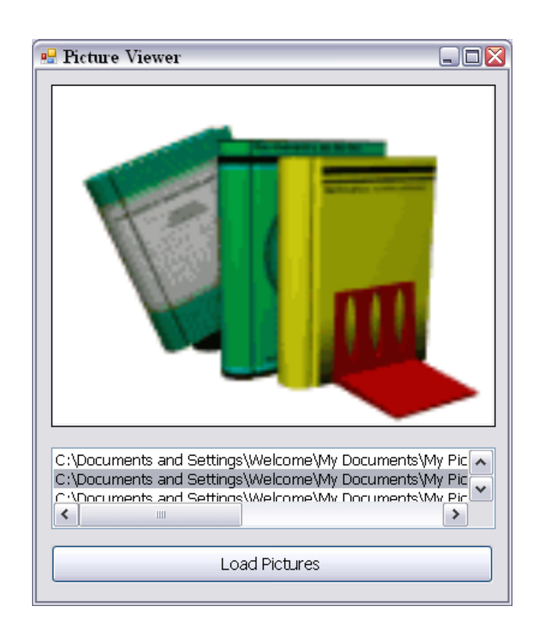
من القائمة Method Name أختر الطريقة SelectedIndexChanged.

في عامل الحدث ListBox1_SelectedIndexChanged أضف التعليمات التالية:

```
Me.PictureBox1.ImageLocation = Me.ListBox1.SelectedItem
```

أنقر المفتاح F5 لتنفيذ البرنامج.

أنقر المفتاح Load Pictures لتظهر قائمة بأسماء ملفات الصور في صندوق القائمة قف، أختر أي من هذه الملفات لعرضه.



شكل 8-74 ك التطبيق أثناء التنفيذ

1.50.3تحسين عمل البرنامج

قد يكون من غير المناسب في بعض الأحيان أن نقوم بعرض أسم الملف كاملاً (أي أسمه محتوياً على المسار)، كما قد تدعو الحاجة أحياناً للتعرف على المسار الرئيسي فقط. و في كلا الحالتين فإن الكائن My.Computer.FileSystem يساعدنا كثيراً، حيث يمكن استخدام الطريقة My.Computer.FileSystem.GetParentPath للتعرف على مسار الملف فقط دون اسم الملف، والطريقة My.Computer.FileSystem.GetName للحصول على أسم الملف فقط دون اسم المسار.

1.51كتابة البيانات في ملف نصي

في هذا القسم سوف نتعلم كيف نقوم بالكتابة داخل ملف نصي باستخدام الكائن My.Computer.FileSystem. وذلك باستخدام الطريقة My.Computer.FileSystem.WriteAllText والتي تقوم بكتابة البيانات في ملف نصي Text File، فإذا كان هذا الملف غير موجود تقوم الطريقة بإنشائه، ويمكن للمستخدم أن يحدد ما إذا كان يريد أنشاء ملف جديد لهذه البيانات المرسلة للملف – في حالة عدم وجود ملف – أو يلزم برنامجه بتعديل ملف موجود عن طريق استخدام True أو False للبارامتر append في الصيغة المستخدمة.

في التدريب التالي نقوم بإضافة نص إلى ملف نصي.

243.أفتح المشروع السابق.

أضف مفتاح Button جديد لواجهة البرنامج.

قم بتعديل خصائص المفتاحين لتصبح كالتالي:

العنصر	الخاصية	القيمة
LoadPictures	Size	32 ,84
	Location	395 ,12
Button2	Name	FavoritesAdd
	Text	Add to Favorites
	Size	32 ,84
	Location	395 ,102

أنقر مرتين فوق المفتاح FavoritesAdd وأضف التعليمات التالية:

```
If PictureBox1.ImageLocation <> "" Then
```

```
' Add the selected picture to the favorites text file.
My.Computer.FileSystem.WriteAllText(FavoritePictures, _
     Me.ListBox1.SelectedItem & ",", True)
End If
```

أضف هذه التعليمات تحت Public Class Form1 مباشرة:

```
Dim FavoritePictures As String =
My.Computer.FileSystem.SpecialDirectories.MyDocuments
& "\FavoritePictures.txt"
```

أضرب F5 لتنفيذ البرنامج.

أختر احد الصور.

أنقر المفتاح Add to Favorites.

أذهب إلى مكان الملف على My Document ثم قم أفتح الملف وتأكد أنه قد كتب الملف فيه.

1.52قراءة البيانات من ملف نصي

في هذا القسم نتعرف على كيفية قراءة البيانات من ملف نصي. يمكن قراءة البيانات النصية من الملف النصي باستخدام الطريقة My.Computer.FileSystem.ReadAllText. لكن في كثير من الأحوال تكون البيانات في الملفات النصية مفصولة بواسطة علامة الفاصلة وهذا النوع من الملفات يطلق عليها الاسم comma limited text files وهو من أنواع الملفات النصية شائعة الاستخدام لاختزان البيانات، لذلك توجد طريقة مباشرة لقراءة البيانات من هذا النوع من الملفات هي الطريقة My.Computer.FileSystem.OpenTextFieldParser.

في التدريب التالي سوف نقوم باختبار ما إذا كان الملف FavoritePictures موجود في المجلد My Document فإذا كان موجود سوف يقوم البرنامج بقراءة البيانات من الملف

244.أفتح المشروع السابق.

أضف مفتاح Button جديد لواجهة البرنامج.

قم بتعديل خصائص المفتاح لتصبح كالتالي:

العنصر	الخاصية	القيمة
Button2	Name	LoadFavorites
	Text	Load Favorites
	Size	32 ,84
	Location	395 ,193

أنقر مرتين فوق المفتاح LoadFavorites وأضف التعليمات التالية:

```
' Clear the picture box and the list box.
Me.ListBox1.Items.Clear()
Me.PictureBox1.ImageLocation = ""
```

ثم أضف التعليمات التالية لتعقب التعليمات السابقة مباشرة، وهذه التعليمات عبارة عن جملة If تختبر ما إذا كان الملف FavoritePictures موجود أم غير موجود. فإذا كان الملف غير موجود فإن البرنامج سوف ينفذ التعليمات اللاحقة، أما إذا كان غير موجود فسيتم تنفيذ التعليمات في الكتلة else.

```
If My.Computer.FileSystem.FileExists(FavoritePictures)
Then
' Add code to read text from a file.
Else
 MsgBox("There is no favorites file yet. Click Load"
 & " Pictures," & vbCrLf & "select a picture, and"  &
" then click Add to Favorites.", MsgBoxStyle.OkOnly,
   "Picture Viewer")
End If
```

استبدل جملة التعليق التالية لـــ Then بالتعليمات التالية، وهي التعليمات التي تستخدم الطريقة OpenTextFieldParsar لقراءة محتويات الملف FavoritePictures.

```
' Open the FavoritePictures text file by using
' OpenTextFieldParser.
Dim MyReader As
Microsoft.VisualBasic.FileIO.TextFieldParser
MyReader = My.Computer.FileSystem.OpenTextFieldParser(
FavoritePictures)
```

```
MyReader.SetDelimiters(",")
```

أ ضف التعليمات التال ية ع قب التعليمات الس ابقة مباشرة، وهذه التعليمات وظيفتها تكرار عملية قراءة البيانات من الملف FavoritePictures.

```
' Using a comma (,) as a delimeter, parse each field
in
' the text file and add it to the list box.
Dim textFields As String() = MyReader.ReadFields()
For Each currentField As String In textFields
  If My.Computer.FileSystem.FileExists(currentField)
Then
     Me.ListBox1.Items.Add(currentField)
  End If
Next
' Close the TextFieldParser.
MyReader.Close()
```

أنقر المفتاح F5.

أنقر المفتاح Load Favorites ليتم تحميل ملفاتك المفضلة فقط.

1.53حذف ملف

في هذا القس م نتعلم ك يف يمكن نا حذف ملف با ستخدام الطري قة My.Computer.FileSystem.DeleteFile. من المنا سب أن يقوم الم مبرمج بس ؤال مستخدم برنامجه حول ما إذا كان متأكد من رغبته في حذف الملف أم لا وبناء على إجابته ينفذ عملية الحذف أو يلغيها.

245.أفتح المشروع السابق.

أضف مفتاح Button جديد لواجهة البرنامج.

قم بتعديل خصائص المفتاح لتصبح كالتالي:

العنصر	الخاصية	القيمة
Button3	Name	DeleteFavorites
	Text	Delete Favorites
	Size	32 ,84
	Location	395 ,283

أنقر مرتين فوق المفتاح DeleteFavorites وأضف التعليمات التالية:

```
' Check that the favorites text file exists.
If My.Computer.FileSystem.FileExists(FavoritePictures)
Then
' Ensure that user wants to delete the favorites text
file.
   If MsgBox("Are you sure you want to send the
favorites" & " file to the Recycle Bin?",
MsgBoxStyle.YesNo, "Delete Favorite Pictures") =
MsgBoxResult.Yes Then
      ' Clear the picture box and the list box.
      Me.ListBox1.Items.Clear()
      Me.PictureBox1.ImageLocation = ""
      ' Delete the favorites file.
My.Computer.FileSystem.DeleteFile(FavoritePictures)
   End If
Else
   MsgBox("The favorites file does not exist.")
End If
```

أنقر المفتاح F5

أنقر وفق المفتاح Delete Favorite

تأكد أن الملف قد تم حذفه من My Document

أسس برمجة الكائنات

كما قدمنا من قبل تعتمد البرامج التي يتم تطويرها باستخدام Visual Basic.NET 2008 على الكائنات Objects بصورة أساسية، هناك من الكائنات ما تقدمها اللغة نفسها مثل عناصر التحكم والنماذج، وهناك الكائنات التي يقوم المبرمج بنفسه ببناءها لأداء حاجات معينة في برنامجه.

يتسم مفهوم الكائن بوجود مفهوم آخر هو مفهوم الفئة Class مصاحب له، وهذا المفهوم يمكن أن نقدمه كالمخطط الهندسي للكائن، فهو يحتوي على البيانات التي من المفترض أن يتعامل الكائن معها، والطرق التي سيستجيب بها، ومجموعة الأحداث التي سيستجيب لها. ومن ثم يتم بناء الكائنات على شاكلته.

1.54ما هي الفئة Class؟

كما قدمنا يمكننا أن ننظر على الفئة على أنها المخطط الهندسي للكائنات، أو في صورة أكثر حرفية ومصداقية على أنها كائن مثالي مجرد سيتم بناء الكائنات على مثاله. ونحن نقوم بالتعامل مع هذه الفئات منذ بداية استخدامنا لـ Visual Basic.NET 2008، فمثلا عنصر التحكم TextBox هو في واقع الأمر عبارة عن فئة، وفي كل مرة نقوم بسحبه إلى سطح النموذج فإننا في الواقع نقوم بإنشاء مثال instance عن هذه الفئة أو من خلال مصطلح مبسط نقوم بإنشاء كائن. وعلى هذا فإن الكائن textbox من الممكن أن نقوم بإنشائه بالطريقة التقليدية التي تعتمد على سحب العنصر ورميه أو نقوم بكتابة التعليمات اللازمة لإنشائه وذلك بصورة بسيطة كما هو مبين في السطر التالي:

```
Dim myTextbox as New TextBox
```

أما أنشاء الفئات واستخدامها فسوف نتعرف عليها بصورة تفصيلية في أجزاء متقدمة من هذا القسم.

1.54.1ماذا بداخل الفئة؟

في القسم ٣.١.٣ تعرفنا على الخصائص والطرق والأحداث، وتعلمنا كيف أن لكل الكائنات خصائصها التي تصفها، والطرق التي تتصرف بها والأحداث التي تستجيب لها.

وبالمثل فإن الفئة لها خصائص وطرق وأحداث (يطلق عليها في بعض الأحيان الأعضاء Members).

سنستخدم مثال بسيط لفهم ما هي الفئة. الحساب البنكي لو قدمناه كفئة اسمها BankAccount فمن الممكن أن يكون لها خصائص مثل رقم الحساب AccountNumber والقيمة الحدية للحساب AccountBlance، وطرق مثل حساب الربح CalcualteInterest، وأحداث مثل تغير القيمة الحدية BalanceChanged. أي حساب بنكي لا بد أن يكون له هذه الخصائص والطرق والأحداث، لكن عند أنشاء كائن BankAccount1 كمثال عن الفئة BankAccount، فإنه سيكون له خصائصه وطرقه وأحداثه المستنسخة عن الفئة BankAccount والتي تختلف عن الخصائص والطرق والأحداث الخاصة بالكائن BankAccount2 المثال عن ذات الفئة.

توجد أحداث / أعضاء يطلق عليها الأحداث الخاصة Private، وهي تلك الأحداث التي يتم تنفيذها داخل الكائن، ولا يتم تنفيذها خارجه. فمثلاً الفئة BancAccount يمكن أن تحتوي على حدث لحساب القيمة الحدية، حيث يقوم البرنامج بقراءة القيمة الحدية لكن لا يقوم بتغيير هذه القيمة.

يمكن للمبرمج أن يقوم بإخفاء الأعضاء داخل الفئة بإعلانهم private أو يمكنه أن يتيحهم للاستخدام خارج الكائن عن طريق إعلانهم Public، كما يمكنه أن يسمح للمرور إلى الخواص ولا يسمح بتغييرها عبر إعلانها خواص للقراءة فقط ReadOnly.

التعليمات التالية تبين الشكل الذي تكون عليه الفئة.

```
Class BankAccount
  Private AccountNumber As String
  Private AccountBalance As Decimal
  Public Sub UpdateBalance()
    ' add code to recalculate balance.
  End Sub
  ReadOnly Property Balance() As Decimal
    Get
      Return AccountBalance
    End Get
  End Property
```

```
End Class
```

1.55أنشاء الفئة

في هذا القسم سوف نخطو خطوة جديدة للأمام حيث سوف نقوم بإنشاء كائن يمثل شخص، ثم نقم بتخزينه لاستخدامه في برامج أخرى. يمكن إنشاء الفئة بإحدى ثلاثة طرق:

بين التعليمات الخاصة بنموذج في أي من تطبيقات النوافذ.

في وحدة نمطية Module مستقلة ولكنها ملحقة بتطبيق النوافذ.

في مشروع مكتبة فئة Class Library مستقل.

1.55.1أنشاء الفئات في المشروع

لعل القارئ قد لاحظ أنه في التدريبات السابقة أنه عند الذقر فوق النموذج مرتين ينفتح محرر التعليمات ويرى القارئ ما يشابه التالي:

```
Public Class Form1
   Private Sub Form1_Load...

   End Sub
End Class
```

في عبارة بسيطة، النموذج Form يعتبر في حد ذاته فئة، لذلك فهي تبدأ بـ Class وتنتهي بـ End Class وأي تعليمات يقوم المبرمج بإضافتها داخل هذه الفئة تعتبر جزء منها. وبالرغم من أن النموذج Form يحتوي على فئة واحدة فقط، إلا أن المبرمج يمكنه أن يضيف المزيد من الفئات عن طريق أن يحررها مبتدئاً بـ Class ومنتهياً بـ End Class وذلك بعد End Class الخاصة بالنموذج Form، وفي تلك الحالة يصبح محرر التعليمات يحتوي على فئتين في آن واحد كما هو مبين فيما يلي:

```
Public Class Form1
   ' Form1 code here
End Class
Public Class MyFirstClass
   ' Your class code here
End Class
```

وتعتبر الفئات المنشأة بهذه الطريقة قاصرة العمل على المشروع الذي يضمها فقط، لـ كن إذا كان الـ مبرمج يخـ طط أن يسـ تخدم هذه الفئة في أكـ ثر من مشروع فيمكـ نه أن يقوم بوضعها في وحدة نمطية كما سيأتي.

1.55.2الوحدات النمطية للفئات

الوحدة النمطية Module للفئة هي عبارة عن ملف تعليمات مستقل يحتوي على واحدة أو أكثر من الفئات، ولأنها ملف مستقل فيمكن استخدامها في أكثر من مشروع. يمكن أنشاء الوحدات النمطية للفئات بطريق من أثنين:

عن طريق إضافة وحدة نمطية إلى مشروع تطبيق، وذلك من خلال النقر على القائمة Project واختيار Add New Item ثم اختيار Class.

عن طريق أنشاء مشروع مكتبة فئة Class Library.

أما في هذا التدريب سوف نقوم بإنشاء مكتبة فئة مستقلة، وذلك متبعين الخطوات التالية:

246.في القائمة File أنقر فوق New Project.

في التبويب Template أختار Class Library.

في الخانة Name أكتب Persons ثم أنقر Ok.

يتم فتح مشروع مكتبة فئة جديد ويعرض حرر التعليمات وحدة نمطية للفئة بعنوان Class1.vb كما هو مبين في شكل 75-9

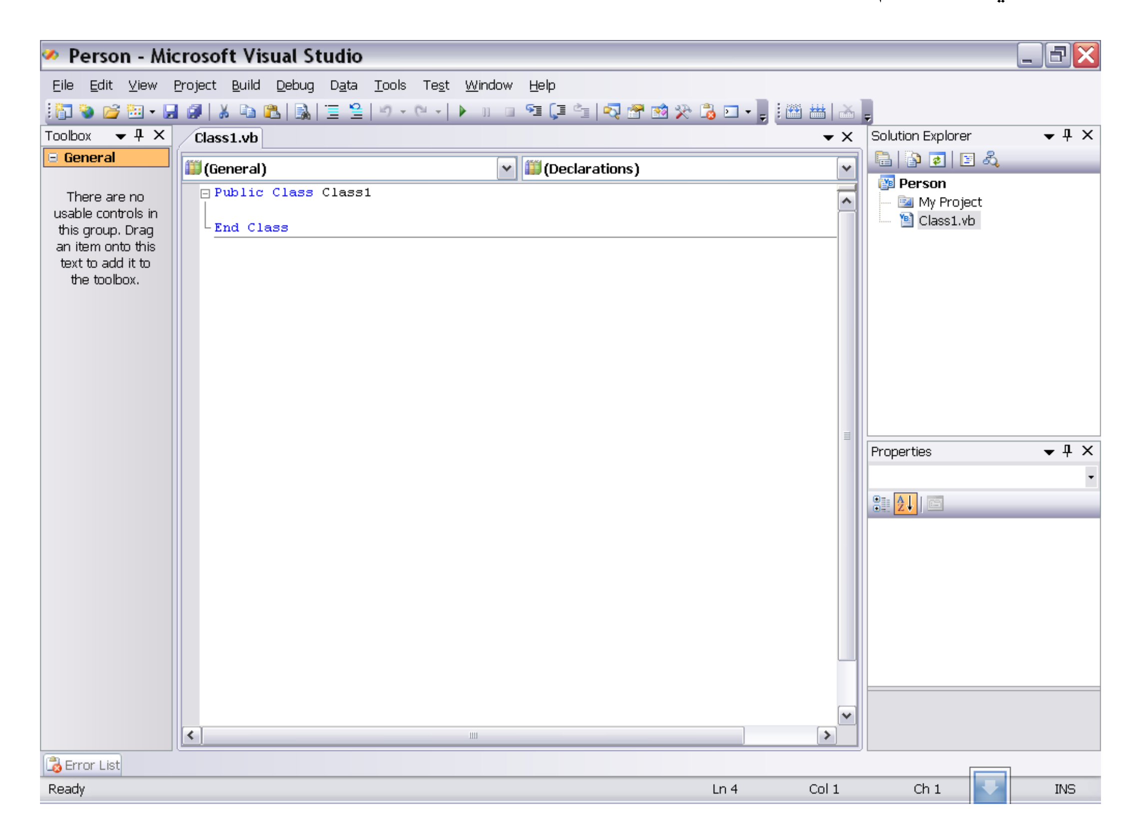
في النافذة Solution Explorer أنقر فوق Class1.vb مستخدماً مفتاح الفأرة الأيمن وأختار Rename ثم قم بتسمية الفئة Person.vb.

أختر من القائمة File الخيار Save All.

في نافذة Save Project أختار Save.

1.56إضافة الخصائص إلى الفئة

في القسم السابق بينا كيف يمكن إنشاء فئة، وقمنا بإنشاء فئة هي الفئة Person التي من المفروض أن تمثل أشخاص عاديين، وحيث أن أي شخص له مجموعة من الخصائص مثل الاسم والسن، فلابد أن تكون للفئة القدرة على تمثيل هذه الخصائص. و هو ما سنحاول تقديمه في هذا القسم.



شكل 9-75: مشروع Class Library في محرر التعليمات

يمكن إضافة الخصائص للفئة بطريقة من اثنتين، إما في صورة حقول Fields مباشرة أو كخصائص في إجراءات Procedure's Properties. وكذلك يمكن إعلان هذه الخصائص كخصائص عامة Public أو خاصة للكتابة فقط WriteOnly أو خواص للقراءة فقط ReadOnly.

1.56.1الحقول وخصائص الإجراءات

يمكن النظر إلى الحقول على أنها متغيرات عامة يمكن الوصول إليها من داخل أو من خارج الفئة. تستخدم الحقول بصورة مثلى لاختزان القيم التي لا تحتاج إلى تحقق validate – على سبيل المثال البيانات من النوع الـ Boolean. في حالة الفئة Person

يمكننا أن نجعل هناك خاصية بالاسم Alive وتكون من النوع Boolean لتبين ما إذا كان الشخص حي أم ميت.

كل ما يحتاجه المبرمج لإضافة خاصية/حقل إلى الفئة التي يعمل علة تطوير ها هو تعليمات إعلان المتغيرات البسيطة كالتالية:

```
Public Alive As Boolean
```

لكن للأسف فإن معظم الخصائص التي تضاف إلى الكائنات تحتاج إلى تحقق ومن ثم فإنه يتم إضافتها كمتغيرات إجرائية. تتكون المتغيرات الإجرائية من ثلاثة أقسام:

إعلان عن متغير خاص Private لاختزان قيمة الخاصية.

إجراء Get ويستخدم لاسترجاع قيمة المتغير،

إجراء Set لتعيين قيمة المتغير.

بهذه الطريقة يمكن أنشاء خاصية كمتغير إجرائي في الفئة، التعليمات التالية تنشئ خاصية Name للكائن Person.

```
Private nameValue As String
Public Property Name() As String
   Get
      Name = nameValue
   End Get
   Set(ByVal value As String)
      nameValue = value
   End Set
End Property
```

في السطر الأول من هذه التعليمات يتم الإعلان عن متغير خاص Private من النوع String باسم nameValue والذي سوف يختزن قيمة الخاصية. المتغير الإجرائي يبدأ مع العبارة Public Property وينتهي مع العبارة End Property.

الإجراء Get يحتوي على التعليمات التي سوف يتم تنفيذها عندما تريد أن تقرأ قيمة الخاصية، فعلى سبيل المثال إذا كنت ترغب في قراءة الخاصية Person.Name فإن التعليمات السابقة سوف تعيد القيمة المختزنة في nameValue.

الإجراء Set يحتوي على التعليمات التي يتم تنفيذ ها عند تخصيص قيمة للمتغير nameValue تستخدم كقيمة يتم تمرير ها كبارامتر لهذا الإجراء. فمثلاً، إذا قمت بكتابة تعليمات بهذا الشكل Person.Name="John"، فإن القيمة John سوف يتم تمرير ها كبارامتر إلى الإجراء Set بينما ستقوم التعليمات الموجودة داخل هذا الإجراء بتخصيص هذه القيمة للمتغير nameValue.

قد تسأل نفسك سؤال لماذا لا أخصص القيمة إلى الحقل Name مباشرة وأقرأها منه مباشرة عوضاً عن كل هذه المتاعب. باختصار هذا يتيح لك أن تتحقق من أن القيم المخصصة للمتغيرات تتوافق مع تصميمك للفئة أم لا، فمثلاً في الحياة اليومية لا يوجد شخص يتكون اسمه أو يحتوي على أرقام، وعلى هذا يمكنك إضافة تعليمات في الإجراء Set لتختبر القيم المدخلة وهل تتوافق مع فرضية أن الأسماء لا تحتوي على أرقام أم لا.

في التدريب التالي سوف نقوم بإضافة مجموعة من المتغيرات الإجرائية للفئة Person.

247.قم بفتح المشروع السابق Person.

قم بإضافة التعليمات التالية عقب العبارة Public Class Person

```
Private firstNameValue As String
Private middleNameValue As String
Private lastNameValue As String
Public Alive As Boolean
```

قم بإضافة التعليمات التالية عقب هذه التعليمات

```
Public Property FirstName() As String
  Get
    FirstName = firstNameValue
  End Get
  Set(ByVal value As String)
    firstNameValue = value
  End Set
End Property

Public Property MiddleName() As String
```

```
   Get
      MiddleName = middleNameValue
   End Get
   Set(ByVal value As String)
      middleNameValue = value
   End Set
End Property

Public Property LastName() As String
   Get
      LastName = lastNameValue
   End Get
   Set(ByVal value As String)
      lastNameValue = value
   End Set
End Property
```

من القائمة File أختر Save All.

1.56.2الخصائص من النوع ReadOnly والنوع WriteOnly

في بعض الأحوال يتم ضبط الخاصية وتخصيص قيمة لها لا تتغير أثناء التشغيل. على سبيل المثال خاصية مثل رقم الموظف لا يجب أن تتغير، حيث يمكن أن تقرأ من خلال البرنامج لكن لا يسمح للبرنامج بتغييرها أبدا.

الكلمة الأساسية ReadOnly تستخدم لتعلم المجمع أن الخاصية للقراءة فقط أي لا يمـ كن تعديلـ ها أثناء التشغـ يل. وإذا حاولت – بالخـ طأ – أثناء كتابـ تك للبرنا مج أن تقوم بتخصيص قيمة للخاصية المعرفة ReadOnly فإن محرر التعليمات سوف يعتبر هذا خطأ.

لإنشاء خاصية من النوع ReadOnly فإنك سوف تقوم بإنشاء متغير إجرائي كما يلي:

```
Private IDValue As Integer
ReadOnly Property ID() As Integer
   Get
      ID = IDValue
```

```
   End Get
End Property
```

لاحظ انه لا وجود للإجراء Set في هذا التعريف.

بالمثل، فإن الكلمة الأساسية WriteOnly تسمح للخاصية أن يتم تخصيصها ولكنها لا تسمح للمبرمج أن يقرأ قيمة هذه الخاصية، على سبيل المثال قد يرغب المبرمج في استخدام هذا النوع من الخواص في تمرير كلمة السر بين برامج مختلفة.

وأيضاً لإنشاء خاصية من هذا النوع يمكنك أن تستخدم تعليمات مثل التالية

```
Private passwordValue As String
WriteOnly Property Password() As String
   Set(ByVal value As String)
      passwordValue = value
   End Set
End Property
```

لاحظ غياب الإجراء Get.

مثل هذا النوع من الخصائص – ReadOnly و WriteOnly – يمكن استخدامه في امثل الطرق عندما يكون المستخدم يرغب في تحويل قيمة إلى أخرى، فعلى سبيل المثال خاصية العمر عندما تقوم بتخصيصها ثم تعاود قراءتها بعد عام فإن القيمة المختزنة فيها سوف تكون خاطئة. ولعلاج هذه المشكلة في الفئة Person يمكن أن نضيف خاصيتين، الأولى BirthDate من النوع WriteOnly والثانية Age من النوع ReadOnly حيث ترجع الخاصية Age الفارق بين قيمة الخاصية الأولى وتاريخ اليوم.

لنفعل ذلك اتبع الخطوات التالية:

248.افتح المشروع السابق Person.

اضف السطر التالي بعد مجموعة الإعلانات (السطور الأربعة الأول بعد Public Class Person)

```
Private birthYearValue As Integer
```

قم بإضافة التعليمات التالية ي نهاية التعليمات التي أضفتها فيما سبق.

```
WriteOnly Property BirthYear() As Integer
```

```
   Set(ByVal value As Integer)
      birthYearValue = value
   End Set
End Property

ReadOnly Property Age() As String
   Get
      Age = My.Computer.Clock.LocalTime.Year -
birthYearValue
   End Get
End Property
```

من القائمة File اختار Save All

1.57إضافة الطرق إلى الفئات

في موضع سابق عرفنا أن لكل فئة مجموعة من الطرق Methods التي تتصرف لها الفئة. في هذا القسم سنتعرف على كيفية تصميم الطرق وإضافتها إلى الفئات.

1.57.1الطرق الخاصة بالفئة

يمكن نقدم الطرق الخاصة بالفئات كأنها روتينات فرعية Subroutines أو وظائف Functions داخل الفئة. فعلى سبيل المثال الفئة Account يمكنها أن تحتوي على طريقة Recalculate لإعادة لتحديث القيمة الحدية للحساب، كما يمكنها أن تحتوي على وظيفة CurrentBalance لاسترجاع هذه القيمة الحدية للحساب. التعليمات التي تمثل هذه الطرق يبينها ما يلي:

```
Public Sub Recalculate()
   ' add code to recalculate the account.
End Sub
Public Function CurrentBalance(ByVal AccountNumber As
Integer) As Double
   ' add code to return a balance.
End Function
```

على حين أن معظم الطلاق داخل الفئات هي من النوع الـ Public، فإن المبرمج قد يرغب في بعض الأحيان أن يقوم بإنشاء طرق تعمل داخل الفئة فقط، فمثلا الفئة person قد

تحتوي على طريقة لحساب عمر العميل، وفي هذه الحالة يقوم المبرمج بتعريف هذه الطريقة بالكلامة الأساسية Private حتى لا يمكن استدعاءها من خارج الطريقة. التعليمات التالية تستخدم لإنشاء طريقة من النوع Private داخل الفئة.

```
Private Function CalcAge(ByVal year As Integer) As
Integer
   CalcAge = My.Computer.Clock.LocalTime.Year - year
End Function
```

يم كن لل مبرمج لاحقاً أن يقوم بتغي ير التعليمات ال تي تس تخدم في حس اب قي مة CalcAge ومع ذلك لن تتأثر التعليمات التي تستخدم الطريقة CalcAge. وهذا الوضع هو التـ عبير الأمـ ثل عن وا حد من أ هم مفاه يم البرمـ جة الموجـ هة للكائنات و هو التغلـ يف Encapsulation.

في التدريب التالي سوف نقوم بإضافة طريقتين إلى الفئة Person، الطريقة الأولى من النوع الـ Public استعادة أسم العميل والثانية وظيفة من النوع Private لحساب سن العميل:

249.افتح المشروع السابق Person.

أ ضف السـ طور التالـ ية عـ قب تعريفات المتغيرات الـ تي أضفنا ها في القسـ م السابق:

```
Public Function FullName() As String
   If middleNameValue <> "" Then
      FullName = firstNameValue & " " & middleNameValue
& " "                                  & lastNameValue
   Else
      FullName = firstNameValue & " " & lastNameValue
   End If
End Function
Private Function CalcAge(ByVal year As Integer) As
Integer
   CalcAge = My.Computer.Clock.LocalTime.Year - year
End Function
```

قم بتعديل خاصية Age لتصبح كالتالي:

```
ReadOnly Property Age() As String
   Get
      ' Age = My.Computer.Clock.LocalTime.Year -
birthDateValue
      Age = CalcAge(birthYearValue)
   End Get
End Property
```

من القائمة File اختار Save All

1.57.2مفهوم الحمل الزائد Overloading

في هذا القسم سوف نتعلم كيف يمكننا إضافة أكثر من شكل من نفس الطريقة في الفئة الواحدة. في التدريب السابق قمنا بإضافة الطرق إلى الفئة Person، في بعض الحالات يجب أن يأخذ المبرمج في اعتباره أن الطريقة لابد أن تستجيب لأنماط مختلفة من المدخلات وبالمثل يمكن أن تنتج أنماط مختلفة من المخرجات. و في هذه الحالة يلجأ المستخدم مفهوم الحمل الزائد Overloading حيث يقوم بتصميم الفئة – أي كتابة التعليمات الخاصة بها – عدة مرات، بحيث تكون كل مرة متسقة مع المدخلات المختلفة المتوقعة.

فيما يلي تعليمات لحمل زائد لطريقة واحدة مرتين بحيث تكون مرة تقبل مدخلات من النوع النصي وفي المرة الثانية تقبل مدخلات من النوع الرقمي الصحيح.

```
Public Sub TestFunction(ByVal input As String)
   MsgBox(input)
End Sub
Public Sub TestFunction(ByVal input As Integer)
   MsgBox(CStr(input))
End Sub
```

عندما يرغب المبرمج في استدعاء هذه الطريقة من داخل برنامجه، يتوقف تنفيذ الطريقة على نوع البيانات المارة إليها، فإذا تم تمرير بيانات نصية يتم تنفيذ الطريقة الأولى ويتم طباعة رسالة تحتوي على البيانات النصية المدخلة للطريقة، أما إذا تم تمرير عدد صحي، فإنه سوف يتم تحويله بداية إلى نصي ثم يتم طباعته في رسالة.

يمكن للمبرمج عمد العديد من الطرق تحت مظلة مفهوم الحمل الزائد.

سنبين في التدريب التالي كيف يمكننا إضافة طريقة بمفهومي حمل زائد في الفئة Person.

250.افتح المشروع السابق Person.

أضف السطور التالية قبل End Class:

```
Public Function MiddleInitial() As String
   MiddleInitial = Left$(middleNameValue, 1)
End Function

Public Function MiddleInitial(ByVal period As Boolean)
As String
   MiddleInitial = Left$(middleNameValue, 1) & "."
End Function
```

من القائمة File اختار Save All

1.58إضافة حدث للفئة

تعلمنا فيما سبق أن لكل كائن خصائص وطرق وأحداث، وعرفنا كيف يمكن تصميم وإضافة الخصائص والطرق إلى الفئة. في هذا القسم نتعرض على أساليب تصميم وإضافة معالجات الأحداث إلى الفئات.

1.58.1الإعلان عن الأحداث وإنشاءها

هناك خطوتين يجب أن يلزمهما المبرمج عند إضافة حدث على الفئة. الأول، يجب أن يقوم بالإعلان Declaration عن الحدث، ثانياً، إنشاء الحدث. إنشاء الحدث يعني أن يعلم المبرمج المجمع عن وجود هذا الحدث. لإضافة حدث إلى الفئة، يقوم المبرمج بإعلان الحدث عن طريق الكلمة الأساسية Event. وهذا يعني أن الكائن المثال على الفئة يمكنه أن يستخدم الحدث المعلن. فعلى سبيل المثال يمكن أن نضيف الحدث AgeCalculated للفئة Persons ثم ننشأ الحدث في الطريقة CalcAge، وعند استدعاء هذه الطريقة، يمكن إضافة تعليمات إضافية مع عملية حساب العمر.

251.افتح المشروع السابق Person.

أضف السطر التالي فوق مجموعة تعريفات الخصائص:

```
Public Event AgeCalculated(ByVal Age As Single)
```

هذا السطر يمثل الإعلان عن الحدث.

استبدل السطور التي تحتويها الطريقة CalcAge بالسطور التالية:

```
Private Function CalcAge(ByVal year As Integer) As
Integer
   Dim Age = My.Computer.Clock.LocalTime.Year - year
   RaiseEvent AgeCalculated(Age)
   CalcAge = My.Computer.Clock.LocalTime.Year - year
End Function
```

من القائمة File اختار Save All

1.58.2تكوين معالج الحدث

إذا كان على المبرمج أن يكتب تعليمات تستجيب لحدث ما – أي يكتب معالج حدث Event handler – عند ذلك يجب الربط ما بين الحدث ومعالج الحدث. ويستخدم في ذلك الكلمات المحجوزة Handles أو Add Handler. الكلامة المحجوزة Event Handler تسمح للمبرمج أن يربط الحدث بالمعالج أثناء التصميم، على حين تسمح الكلمة المحجوزة Handles بربط الحدث بمعالج الحدث أثناء التشغيل. هذه الكلمة المحجوزة Handles يمكن إضافتها إلى نهاية أي روتين فرعي لتقوم بنفس عمل الحدث. فعلى سبيل المثال، يمكن أن نضيف حدث باسم AgeCalculated الذي يمكنه اخذ أي عدد صحيح، وكذلك الروتين الفرعي سيقوم بأخذ هذه القيمة الرقمية الصحيحة.

```
Private Sub person1_AgeCalculated(ByVal Age As Integer)
Handles person1.AgeCalculated
```

الكائن person1 لابد من إنشاءه باستخدام الكلمة المحجوزة WithEvents ومن ثم يمكن استخدام معالج الحدث AgeCalculated.

1.59اختبار الفئات

فيما سبق قمنا بإنشاء الفئة Person وأضفنا إليها الخصائص والطرق والأحداث، في هذا القسم سوف نتعلم كيف يمكننا إنشاء كائن مثال للفئة Person وذلك من أجل أختبار الفئة التي قمنا بتصميمها.

1.59.1إنشاء كائن مثال على الفئة

بالر غم من عد إدراكنا لهذه الحقيقة، فإننا كنا نتعامل مع فئات طوال التدريبات السابق، فالنماذج وعناصر التحكم ما هي إلا فئات في واقع الأمر، وعندما نقوم بسحب أي من عناصر التحكم ونرميه فوق النموذج، فإننا نقوم بإنشاء مثال على فئته.

يمكننا أن ننشئ مثال عن أي كائن عن طريق التعليمات وذلك باستخدام الكلمة المحجوزة New. فمثلاً عوضاً عن سحب المفتاح Button ورميه يمكننا أن نستخدم التعليمات التالية:

```
Dim aButton As New Button
```

وحتى يمكننا أن نختبر الفئة Person فإنه سيكون علينا إنشاء مشروع جديد ثم عمل إشارة مرجعية Reference للفئة Person ومن ثم نستطيع استخدامه في المشروع. ولذلك سوف نقوم بالإجراءات التالية:

252.افتح المشروع Person.

من القائمة File أنقر Ad ثم أنقر New Project.

أختار Window Application.

في الخانة Name أعطي مشروعك الجديد أسم هو PersonTest ثم أنقر المفتاح Ok.

في النافذة Solution Explorer أختار المشروع PersonTest.

من القائمة Project أنقر فوق Set as StartUp Project.

من القائمة Project اختار Add Reference.

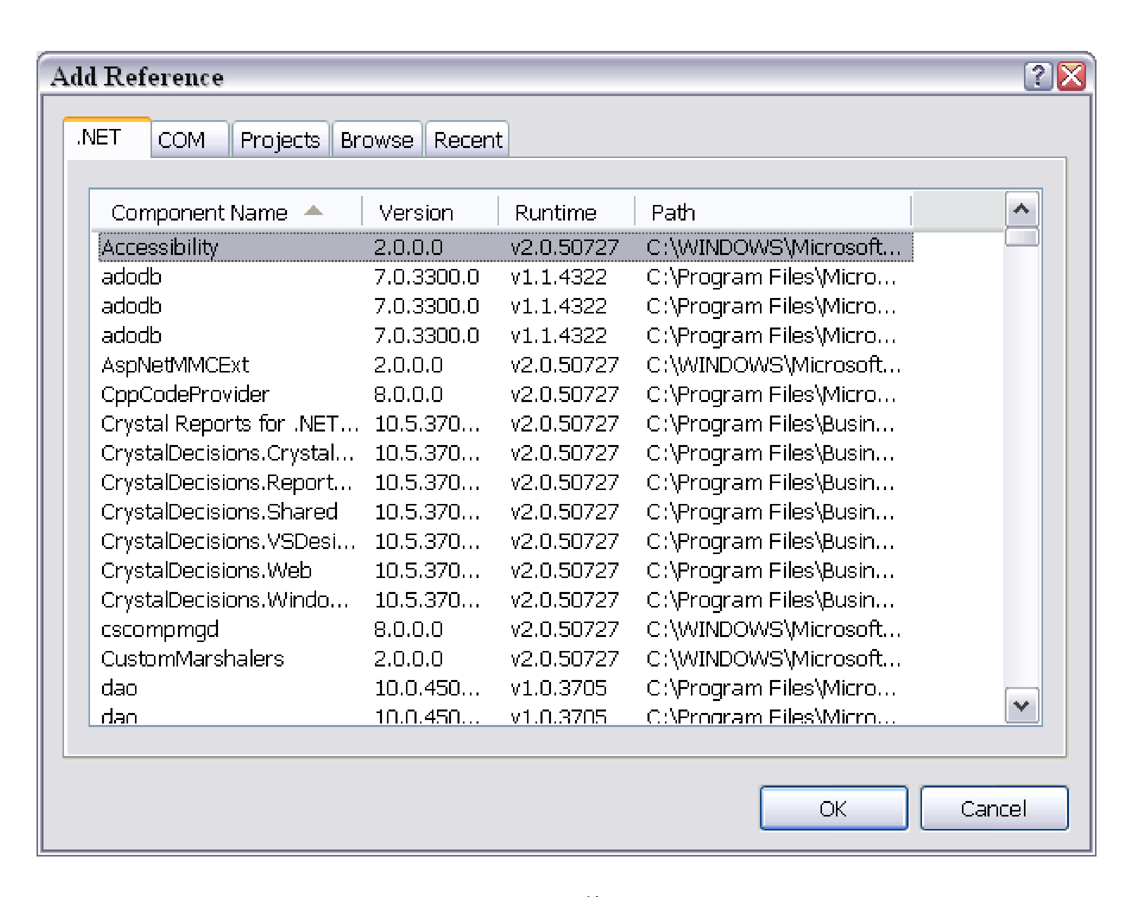
تظهر نافذة Add Reference المبينة في شكل 9-76.

في النافذة Add Reference أنقر فوق التبويب Projects.

أختر Person ثم أنقر المفتاح Ok.

أنقر مرتين فوق النموذج Form1 ليظهر محرر التعليمات.

أضف التعليمات التالية عقب Public Class مباشرة.



شكل 9-76: النافذة Add Reference

```
Dim person1 As New Person.Person
```

هذا السطر يعلن عن مثال جديد عن الفئة Person. وقبل أن تسال لماذا استخدمنا كلمة person مرتين بعد الكلمة المحجوزة New، فالأولى تمثل الوحدة Person.vn النمطية للفئة، بينما الثانية تمثل الفئة Person داخل هذه الوحدة النمطية.

من القائمة File أختار Sava All.

1.59.2اختبار الفئة

في المرحلة الثانية نقوم بإضافة واجهة استخدام User Interface وتعليمات تستخدم الفئة Person. سوف نقوم بإضافة صناديق نصوص حتى يمكن للمستخدم أن يخصص قيم لخصائص الفئة – طبعا الخصائص التي ليست من النوع ReadOnly – وصندوق تأشير لإدخال خاصية Alive، ومفاتيح لاختبار الفئات.

253.من النافذة Solution Explorer أنقر فوق النموذج Form1 ثم أختر Designer.

254. قم بسحب أربعة صناديق نصوص TextBox و صندوق تأشير CheckBox ومفتاحين Button وأرمهم فوق النموذج Form1.

255.أنقر فوق Button1 ثم أنتقل إلى نافذة الخصائص، قم بضبط قيمة الخاصية Text لتصبح Update.

256.أنقر فوق Button2 ثم أنتقل إلى نافذة الخصائص، قم بضبط قيمة الخاصية Text لتصبح Full Name.

257.انقر مرتين فوق المفتاح Update ثم أكتب في معالج الحدث Button1_Click التعليمات التالية:

```
With person1
   .FirstName = Textbox1.Text
   .MiddleName = Textbox2.Text
   .LastName = Textbox3.Text
   .BirthYear = Textbox4.Text
   .Alive = CheckBox1.Checked
End With
```

لعل القارئ يلاحظ أنه عندما يقوم بكتابة التعليمات فإن الخاصية IntelliSense تتصرف مع الفئة Person كأنها من الفئات المتعارف عليها في Visual Basic.NET 2008.

أنقر فوق المفتاح الثاني وأكتب التعليمات التالية في معالج الحدث Button2_Click:

```
' Test the FullName method.
MsgBox(person1.FullName)

' test the Age property and CalcAge method.
MsgBox(CStr(person1.Age) & " years old")

' Test the Alive property.
If person1.Alive = True Then
   MsgBox(person1.FirstName & " is alive")
Else
   MsgBox(person1.FirstName & " is no longer with us")
End If
```

أنقر فوق المفتاح F5 لتنفيذ البرنامج.

في الصندوق الأول قم بكتابة أسمك الأول.

في الصندوق الثاني أكتب اسمك الأوسط.

في الصندوق الثالث أكتب أسم العائلة.

في الصندوق الرابع اكتب عام ميلادك.

أنقر فوق الصندوق التأشير.

أنقر فوق المفتاح Update لتزويد الفئة بالخصائص.

أنقر فوق المفتاح Full Name.

تظهر ثلاثة رسائل الأولى عن الاسم الكامل والثانية عن العمر والثالثة عن الحالة الحياتية.

من القائمة File أختر Save All.

1.59.3اختبار التحميل الزائد للفئات

258.أضف مفتاحين جديدين للنموذج.

أضبط الخاصية text للمفتاح الأول لتصبح With وللمفتاح الثاني لتصبح Without.

أنقر مرتين فوق المفتاح With وأضف التعليمات التالية للحدث Button3_Click:

```
MsgBox(person1.FirstName & " " &
person1.MiddleInitial(True) & " " & person1.LastName)
```

أنقر مرتين فوق المفتاح Without وأضف التعليمات التالية للحدث Button4_Click:

```
MsgBox(person1.FirstName & " " & person1.MiddleInitial
& " " & person1.LastName)
```

أنقر فوق المفتاح F5 لتنفيذ البرنامج.

في الصندوق الأول قم بكتابة أسمك الأول.

في الصندوق الثاني أكتب اسمك الأوسط.

في الصندوق الثالث أكتب أسم العائلة.

في الصندوق الرابع اكتب عام ميلادك.

أنقر فوق الصندوق التأشير.

أنقر فوق المفتاح Update لتزويد الفئة بالخصائص.

أنقر فوق المفتاح With.

تظهر رسالة تحتوي على أسمك وعلامة بعد الاسم الأوسط

أنقر فوق المفتاح Without.

تظهر رسالة تحتوي على أسمك وبدون علامة بعد الاسم الأوسط

من القائمة File أختر Save All.

1.59.4 اختبار عامل الحدث

259. أضف العبارة التالية قبل الإعلان عن المتغير person (أي بعد Public Class).

```
WithEvents person1 As New Persons.Persons
```

أضف الإجراء التالي إلى النموذج

```
Private Sub person1_AgeCalculated(ByVal Age As Integer)
Handles person1.AgeCalculated
 If Age > 18 Then
  MsgBox("You have been over 18 for " & Age - 18 & "
years.")
 Else
  MsgBox("You will be 18 in " & 18 - Age & " years")
 End If
End Sub
```

أنقر فوق المفتاح F5 لتنفيذ البرنامج.

في الصندوق الأول قم بكتابة أسمك الأول.

في الصندوق الثاني أكتب اسمك الأوسط.

في الصندوق الثالث أكتب أسم العائلة.

في الصندوق الرابع اكتب عام ميلادك.

أنقر فوق الصندوق التأشير.

أنقر فوق المفتاح Update لتزويد الفئة بالخصائص.

أنقر فوق المفتاح Full Name.

إذا كان عمرك تجاوز الثامنة عشر، تظهر رسالة تعلمك أنك تجاوزت الثامنة عشر، وتليها رسالة تبين بكم تزيد عن الثامنة عشر، أما إذا كنت دون الثامنة عشر، تظهر رسالة تبين الفارق بين سنك والثامنة عشر، ثم تظهر رسالة تبين أنك دون الثامنة عشر.

من القائمة File أختر Save All.

1.60الوراثة Inheritance

الوراثة تمثل خاصية مهمة في التعامل مع الفئات. وحتى نقربها من الذهن، لنتخيل السيارات كفئة، تتفق جميع السيارات في الخصائص والوظائف والأحداث، لذلك ننظر لها

كفئة واحدة، لكن هناك بعض السيارات التي لها خصائص فريدة كتلك السيارات التي لها سقف قابل للطي، هذه السيارات تتفق مع جميع السيارات الأخرى في الخصائص والوظائف والأحداث، لكنها تزيد عنها في سقفها القابل للطي. إذا نظرنا إلى هذا الموقف من وجهة نظر الفئات، فنحن لن نكون في حاجة أن ننشأ فئة جديدة بالكامل لتمثل مجموعة السيارات ذوات السقف القابل للطي، بل سوف ننشأ فئة متفرعة عن الفئة الأساسية للسيارات ترث عن الفئة الأم جميع الخصائص والطرق والأحداث المعرفة في هذه الفئة الأم، إضافة إلى مجموعة جديدة من الخصائص والأحداث والوظائف التي تمثل تميز هذه الفئة عن الفئة الأم. يطلق على هذه المقاربة في البرمجة اسم الوراثة Inheritance.

1.60.1الوراثة من فئة موجودة

الكلمة المحجوزة Inherits تستخدم لإعلان فئة جديدة يطلق عليها أسم الفئة المشتقة Derived Class ترث فئة موجودة يطلق عليها اسم الفئة الأساسية Base Class. التعليمات التالية تبين كيف يمكن استخدام الكلمة المحجوزة Inherits:

```
Class DerivedClass
   Inherits BaseClass
End Class
```

الفئة الجديدة DerivedClass يمكن إنشاء كائنات عنها الحين، و سوف يكون لكائناتها جميع خصائص ووظائف وأحداث الفئة BaseClass. لنضرب مثلاً حتى يمكننا تصور الفكرة التي نناقشها. فلنتصور أننا نريد أن ننشأ فئة تمثل لاعبي كرة القدم، لكن لاعبي كرة القدم لهم كل الخصائص والأحداث والوظائف لأفراد الفئة Person التي أنشأنها من قبل. لذلك فلا داعي لأن نقوم بإنشاء فئة خاصة جديدة Players، لكن يمكننا أن ننشأ هذه الفئة كفئة مشتقة أو منحدرة من الفئة Person.

حتى نقوم بهذا العمل لنتبع الإجراءات التالية:

260.أفتح المشروع Person.

أنقر فوق المشروع Person في النافذة Solution Explorer.

من القائمة Project أختار Add Class.

في النافذة Add New Item أكتب Players في الخانة Name.

انقر المفتاح Add.

تظهر وحدة نمطية جديدة.

في نافذة محرر التعليمات أضف التعليمات التالية:

```
Inherits Persons
```

أضف بعد هذه السطر التعليمات التالية التي تعرف خاصيتين جديدتين:

```
Private numberValue As Integer
Private positionValue As String
Public Property Number() As Integer
  Get
    Number = numberValue
  End Get
  Set(ByVal value As Integer)
    numberValue = value
  End Set
End Property
Public Property Position() As String
  Get
    Position = positionValue
  End Get
  Set(ByVal value As String)
    positionValue = value
  End Set
End Property
```

من القائمة File أختر Save All.

1.60.2اختبار الفئة المشتقة

261.من القائمة File أختر Add ومنها اختار New Project.

في النافذة Add New Project أختار Windows Application.

في الخانة Name أكتب PlayerTest.

انقر Ok.

يظهر نموذج جديد

أنقر فوق المشروع PlayerTest في نافذة Solution Explorer.

من القائمة Project أختار Set as Stratup Projetc.

من القائمة Project أختار Add Reference.

في النافذة Add Reference انقر فوق التبويب Projects وأختار Persons ثم Ok.

أنقر مرتين فوق النموذج لتفتح محرر التعليمات، أكتب التعليمات التالية تحت Public Class Form1.

```
Dim player1 As New Persons.Players
Dim player2 As New Persons.Players
```

هذين السطرين يعلنان عن كائنين جديدين.

أنقر مرتين فوق النموذج الموجود في المشروع PlayerTest لتظهر نافذة محرر التعليمات.

أضف التعليمات التالية لعامل الحدث Form1_Load:

```
With player1
   .FirstName = "Andrew"
   .LastName = "Cencini"
   .Number = 43
   .Position = "Shortstop"
End With
With player2
   .FirstName = "Robert"
   .LastName = "Lyon"
   .Number = 11
   .Position = "Catcher"
End With
```

أضف مفتاحين Button للنموذج.

أختر المفتاح الأول وقم بضبط الخاصية Text له لتصبح At Bat.

أختر المفتاح الأول وقم بضبط الخاصية Text له لتصبح On Deck.

أنقر مرتين فوق المفتاح At Bat لتظهر نافذة محرر التعليمات.

أضف التعليمات التالية للحدث button1_Click:

```
MsgBox(player1.Position & " " & player1.FullName & ",
#" &  CStr(player1.Number) & " is now at bat.")
```

لاحظ أننا استخدمنا الطريقة Full Name بالرغم أنها لم يتم أضافتها إلى الفئة Player وذلك لأنها موروثة عن الفئة Person الفئة الأساس التي انحدرت منها الفئة Players.

أضف للحدث Button2_Click التعليمات التالية:

```
MsgBox(player2.Position & " " & player2.FullName & ",
#" & CStr(player2.Number) & " is on deck.")
```

من القائمة File أختر Save All.

أنقر المفتاح F5 لتنفيذ البرنامج، ثم أختبر كل مفتاح على حده.

1.60.3 تجاوز الأعضاء Overriding Members

سبق أن أشرنا على كلمة عضو Member تعني مكونات الفئة من خصائص وطرق وأحداث. في هذا القسم سنتعلم كيف يمكن أن نتجاوز أحد أعضاء الفئة الأساس. ونعني بالتجاوز تغيير عمل طريقة أو خاصية منحدرة عن الفئة الأساس في الفئة المشتقة.

فمثلاً يمكن أن ننشئ فئة جديدة للشاحنات Trucks منحدرة من فئة السيارات Cars، ولكن في الشاحنات تكون المحركات تعمل بالديزل، لذا فغن خصائصها سوف تختلف عن تلك العاملة في السيارات العادية، لذلك يلزم هنا تغيير الطريقة التي تصف عمل المحرك أو في عبارة أكثر مهنية تجاوز Override هذه الطريقة.

افتراضياً، لا يمكن تجاوز الطرق والخصائص المنحدرة عن الفئة الأساس، وحتى يتمكن المبرمج من تنفيذ تجاوز عن عضو ما في الفئة المشتقة، لابد أن يكون هذا العضو

قا بل للتجاوز Overridable في الفئة الأ ساس وذلك بإعلا نه كذلك بوا سطة الكل مة المحجوزة Overridable كما هو مبين في التعليمات التالية:

```
Public Overridable Property EngineType As String
Public Overridable Sub StartEngine(ByVal EngineType As String)
```

وعند إنشاء الفئة المشتقة، فإن الأعضاء الذين أشير إليهم كأعضاء قابلة للتجاوز في الفئة الأ ساس، يم كن تعديل هم با ستخدام الكل مة المحجوزة Overrides كما هو مبين في التعليمات التالية:

```
Public Overrides Property EngineType As String
Public Overrides Sub StartEngine(ByVal EngineType As String)
```

سوف نقوم في التدر يب التالي باختبار هذه المفاهيم عن طر يق تعد يل الطري قة FullName لتقوم بإرجاع الاسم بدون الاسم الأوسط ومضافاً إليه رقم اللاعب. لنقوم بهذا لنتبع الإجراءات التالية:

262. في النافذة Solution Explorer أختر person.vb ثم أختر Code من القائمة View.

في نافذة محرر التعليمات قم بتعديل تعريف الطريقة FullName كما يلي:

```
Public Overridable Function FullName() As String
```

263. في النافذة Solution Explorer أختر Player.vb ثم أختر Code من القائمة View.

264. أضف التعليمات التالية إلى تعليمات هذه الفئة:

```
Public Overrides Function FullName() As String
   FullName = FirstName & " " & LastName & ", #" & numberValue
End Function
```

265. في النافذة Solution Explorer أختر PlayerTest ثم أختر Code من القائمة View.

266. في نافذة محرر التعليمات، قم بتعديل عامل الحدث Button1_Click ليصبح كما يلي:

```
MsgBox(player1.Position & " " & player1.FullName & " is now at bat.")
```

267. في نافذة محرر التعليمات، قم بتعديل عامل الحدث Button2_Click ليصبح كما يلي:

```
MsgBox(player2.Position & " " & player2.FullName & " is on deck.")
```

من القائمة File أختر Save All.

أنقر المفتاح F5 لتنفيذ البرنامج، ثم أختبر كل مفتاح على حده.

1.61 استخدام المجموعات في إدارة الكائنات المتعددة

في قسم سابق من هذا الكتاب تعرضنا لاستخدام المصفوفات لإدارة مجموعات من المتغيرات. بنفس هذه الطريقة يمكنك ان يتم إدارة مجموعة من الكائنات باستخدام المصفوفات، لكن Visual Basic.NET 2008 يعرض طريقة أخرى لإدارة مجموعات الكائنات من خلال كائن مخصوص هو كائن "المجموعة Collection" والممكن استخدامه لاختزان أو استرجاع مجموعة كائنات في آن واحد.

وكما في المصفوفات، فإن كل عنصر في المجموعة له دليل Index يمكن الوصول إلى هذا العنصر/الكائن من خلاله. إضافة على ذلك فعن كل عنصر في المجموعة له مفتاح Key و هو عبارة عن قيمة حرفية يمكن أن تستخدم لتعيين هذا العنصر. ميزة استخدام المفتاح أن المبرمج لا يحتاج أن يستخدم الدليل Index للوصول إلى العنصر بل عوضاً عن ذلك يمكنه أن يستخدم أسم ذي معنى.

1.61.1 إنشاء المجموعة

المجموعات تعتبر مهمة للمبرمج الذي يستخدم عدد كبير من الأمثلة الدالة على فئة معينة، فعلا سبيل المثال بالنسبة للفئة Players يحتاج المبرمج على عدد قد يبغ عشرين مثال على هذه الفئة عند عمل برنامج يشير إلى فريق كرة قدم. لحل مثل هذه المسالة يلجأ

المبرمج إلى استخدام المجموعات، وأول خطوة حتى يقوم باستخدام المجموعات هي إنشاء مجموعة جديدة، ولأداء هذا المهمة يحتاج إلى تعليمات مثل التالية:

```
Dim baseballTeam As New Collection
```

بمجرد إنشاء المجموعة يمكن للمبرمج أن يقوم باستخدام الطريقة Add لإضافة كائن إلى المجموعة والطريقة Remove لحذف كائن من هذه المجموعة. عند إضافة كائن إلى المجموعة باستخدام الطريقة Add يحتاج المبرمج إلى تخصيص الكائن الذي سوف يضاف إلى المجموعة ويقوم بتحديد القيمة الحرفية المستخدمة كمفتاح كما هو مبين في التعليمات التالية:

```
baseballTeam.Add(playerObject, "Player's Name")
```

أما عند حذف الكائن من المجموعة فعلى المبرمج أن يستخدم المفتاح لتعيين الكائن المطلوب حذفه من المجموعة كما هو مبين في التعليمات التالية:

```
baseballTeam.Remove("Player's Name")
```

في التدريب التالي سوف نقوم بإضافة كائنين من الفئة Players ثم نقوم بإنشاء مجموعة ونضيف إليها اللاعبين وسوف نستخدم موقع Position اللاعب في الفريق بمثابة المفتاح الخاص بالكائن في المجموعة.

268.افتح المشروع السابق

من النافذة Solution Explorer أختار المشروع PlayerTest ثم أختار النموذج Form1.vb ثم أنقر فوق مفتاح Code.

أضف التعليمات التالية بعد الإعلان عن لكائن player2.

```
Dim player3 As New Persons.Players
Dim player4 As New Persons.Players
Dim team As New Collection
```

أضف التعليمات التالية للحدث Form1_Load:

```
With player3
   .FirstName = "Eduardo"
   .LastName = "Saavedra"
   .Number = 52
```

```
   .Position = "First Base"
End With

With player4
   .FirstName = "Karl"
   .LastName = "Jablonski"
   .Number = 22
   .Position = "Pitcher"
End With

team.Add(player1, player1.Position)
team.Add(player2, player2.Position)
team.Add(player3, player3.Position)
team.Add(player4, player4.Position)
```

من النافذة Solution Explorer أختار PlayerTest ثم أختار Form1.vb وانقر فوق Designer.

أضف إلى النموذج Form1 عنصر تحكم من النوع ComboBox.

في النافذة Properties اختار الخاصية Items ثم انقر فوق ....

أكتب القيم التالية في المحرر الذي سوف يظهر

```
Keeper
Left Wing
Right Wing
```

Attacker

أنقر مرتين فوق الـــ ComboBox لتفتح محرر التعليمات الخاص به، ثم أضف التعليمات التالية للحدث ComboBox1_SelectedIndexChanged:

```
Dim SelectedPlayer As Persons.Players
SelectedPlayer = team(ComboBox1.SelectedItem)
MsgBox("Playing " & ComboBox1.SelectedItem & " is " &
SelectedPlayer.FullName & "!")
```

أنقر F5 لتنفيذ البرنامج. أختر من الـــ ComboBox موقع اللاعب، تعرض بيانات اللاعب الذي يحتل هذا الموقع في رسالة.

1.61.2الحلقة For Each Next

في التدر يب السـ ابق قمـ نا بإضا فة ق يم الـ ـ Position يدوياً في عنصـ ر التحـ كم ComboBox. هذه الطريقة قد تبدو سهلة لـكن عـند التعا مل مع المجموعات الـتي تمـثل كائنات حقيقية تصبح هذه الطريقة مستحيلة و غير عملية بالمرة. لكن الطريقة المثلى للتعامل مع إضافة الكائنات في المجموعات هي استخدام حلقة مخصصة لذلك هي حلقة For Each Next. التعليمات التالية تستخدم هذه الحلقة لأداء مهمة إضـافة الكائنات إلى مجموعة:

```
Dim player As Persons.Players
For Each player In team
   ComboBox1.Items.Add(player.Position)
Next
```

في هذه التعليمات يلعب الكائن player دور العداد في حلقة For – Next التقليدية، فتكون التعليمات تقول لكل كائن player في المجموعة team نفذ عملية إضافة.

أسس برمجة عناصر التحكم

في القسم السابق تناولنا كيف يمكن برمجة الفئات ثم مجها في أكثر من تطبيق، وهو ما يقلل المجهود والوقت اللازم لعملية تطوير التطبيق. عناصر التحكم هي في وقاع الحال مجرد فئات يمكن إعادة استخدامها في العديد من المشروعات. في كثير من الأحوال يجد المستخدم نفسه يحتاج أن يقوم بتصميم ذات واجهة الاستخدام في أكثر من مشروع، كأن يستخدم عنصر تحكم من الفئة TextBox لتلقي الاسم الأول وآخر من نفس الفئة لتلقى اسم العائلة، ثم يقوم بدمج هذين الاسمين للحصول على الاسم الكامل. في مثل هذه الحالة يكون من الأفضل لو أن هناك عنصر تحكم يستطيع المبرمج إعادة استخدامه في تطبيقاته دون أن يعيد كتابته كل مرة.

من هذا المدخل نقدم مفهوم برمجة عناصر التحكم حيث يمكن النظر إلى عناصر التحكم كأنها فئات تقوم بإنشاء كائنات مرئية. معظم عناصر التحكم التي يقوم بتطويرها المبرمجين في واقع الأمر عبارة عن دمج لعدد من عناصر التحكم القياسية المتوفرة في Visual Basic.NET 2008.

في هذا القسم سوف نتعرف على الأساليب والأدوات اللازمة لإنشاء عناصر تحكم خاصة بنا عن طريق دمج عدد من العناصر القياسية.

1.62استخدام User Control Designer

في هذا القسم سوف نتعلم كيف يمكننا أن ننشأ عناصر التحكم خاصتنا باستخدام الأداة User Control Designer. في القسم ٩ تعلمنا ان ننشئ فئة باستخدام قالب المشروع Class Library، عنصر التحكم هو عبارة عن فئة يمكن أن نرثها، وكما عناصر التحكم القياسية المتوفرة في Visual Basic.NET 2008، فإن عناصر التحكم التي يقوم المبرمج بتصميمها يمكن أن يضيفها إلى النماذج أثناء التصميم لتظهر عند تنفيذ البرنامج.

عندما تقوم بتصميم واجهة تطبيقك، فإن النموذج هو المكان الذي سوف تقوم بتجميع عناصر تحكم فوقه وتقرر كيف سوف تبدو هذه العناصر. وبالمثل فإن الأداة User Control Designer تساعدك على تصميم عنصر التحكم خاصتك وكيف يبدو.

عنصر التحكم كأي فئة أخرى لكن يمكن أضافته إلى صندوق الأدوات Toolbox ويمكن وضعه فوق النموذج. وعلى حين أن الوحدة النمطية للفئة Class Module تحتوي فقط على تعليمات، فإن عنصر التحكم يحتوي على تعليمات وشكل. الأداة User Control Designer تماثل النموذج المستخدم في تصميم الواجهات، حيث له خصائص للتحكم في شكل وسلوك عناصر التحكم التي يحتويها.

تختلف الطريقة التي يمكن أن تستخدم إنشاء عنصر التحكم تبعاً لإصداره Visual basic المستخدمة، فـ Visual Basic 2008 يحتوي على مشروع هو Windows Control Library، لكن في Visual Basic Express لابد من أنشاء مشروع Class Library في البداية ثم إضافة قالب User Control.

1.62.1إنشاء عنصر التحكم في Visual Basic Express

لإنشاء عنصر تحكم في Visual basic Express نتبع الخطوات التالية:

269.من القائمة File اختر New Project.

من التبويب Template اختر Class Library ثم أنقر المفتاح Ok.

من القائمة Project أختار Add User Control.

في النافذة Add New Item اختر User Control.

في الخانة Name اكتب NamesControl ثم أنقر Add.

يتم إضافة عنصر تحكم User Control جديد إلى المشروع ويتم فتح نافذة User Control Designer.

من النافذة Solution Explorer انقر بمفتاح الفأرة الأيمن فوق Class1.vb ثم أنقر Delete وانقر Ok.

من النافذة File اختر Save All.

في النافذة Save project اكتب اسم المشروع NamesUserControl ثم أنقر Save.

1.62.2إنشاء عنصر التحكم في Visual Basic 2008

لإنشاء عنصر تحكم في Visual basic 2008 نتبع الخطوات التالية:

270.من القائمة File أختر New project.

من التبويب Templates اختار Windows Control Library.

في الخانة Name أكتب NamesControl ثم أنقر Ok.

يضاف قالب User Control إلى المشروع وتنف تح النافذة User control Designer.

من النافذة File اختر Save All.

في النافذة Save project اكتب اسم المشروع NamesUserControl ثم أنقر Save.

1.63إضافة عناصر التحكم القياسية على عنصر التحكم المصمم

كما سبق أن ذكرنا، فإن أشهر انواع عناصر التحكم المخلقة هي تلك المدمجة من عناصر تحكم قياسية. يمكن أضافة عناصر التحكم القياسية إلى قالب User Control بجرها من صندوق الأدوات ثم رميها فوق القالب.وبمجرد وضع عناصر التحكم القياسية فوق القالب يمكن التحكم في حجمها ووضعها من خلال النافذة Properties. في التدريب التالي سوف نقوم بإضافة عنصر تحكم Label لعرض الاسم الكامل وثلاثة عناصر تحكم من النوع Textbox لإدخال الاسم الأول والأوسط واسم العائلة.

حتى نقوم بهذا سوف نتبع الخطوات التالية:

271.أفتح المشروع NamesUserControl.

في النافذة Solution Explorer اختار NamesControl.vb ثم من القائمة View اختار Designer.

قم بسحب عنصر تحكم Label من صندوق الأدوات وارمه فوق قالب التصميم.

في النافذة Properties قم بتغيير الخاصية Name لتصبح FullName.

من صندوق الأدوات ا سحب ثلا ثة عنا صر تحـ كم من النو ع TextBox وارمهم فوق قالب التصميم.

من النافذة Properties قم بتغيـ ير الأ سماء لتصـ بح FirstName و MiddleName و LastName.

من القائمة File أختار Save All.

1.64إضافة التعليمات إلى عنصر التحكم المنشأ

في هذا القسم نتعلم كيف نضيف التعليمات لعنصر التحكم لعرض الاسم بالكامل. كما عناصر التحكم القياسية، لعناصر التحكم المنشأة خصائص وطرق وأحداث. ويمكن للمبرمج أن يقوم بكتابة التعليمات التي سوف يتم تنفيذها عن الاستجابة للأحداث، وما هي الخصائص التي يمكن مستخدم عنصر تحكمك أن يتعامل معها.

1.64.1الأحداث الخاصة بعنصر التحكم المنشأ

حـ تى يكون عنصـ ر التحـ كم الذي قـ مت بتصـ ميمه ذي منفـ عة، يـ جب أن تضـ يف له تعليمات للتعامل مع أحداثه. ولا تختلف كتابة تعليمات معاملات الحدث لعنصر التحكم عن مثيلتها في واجهات التطبيقات التي قمنا بإنشائها في أقسام سابقة من هذا الكتاب.

في المثال التالي، سوف تقوم بإضـا فة تعليمات تقوم بتحد يث قيـ مة الخا صية Text لعنصر التحكم FullName لتضم القيم النصية المدرجة في عناصر التحكم FirstName و MiddleName و LastName بمجرد إدخال هذه القيمة وذلك من خلال عامل الحدث TextChanged.

272.أفتح المشروع NamesUserControl.

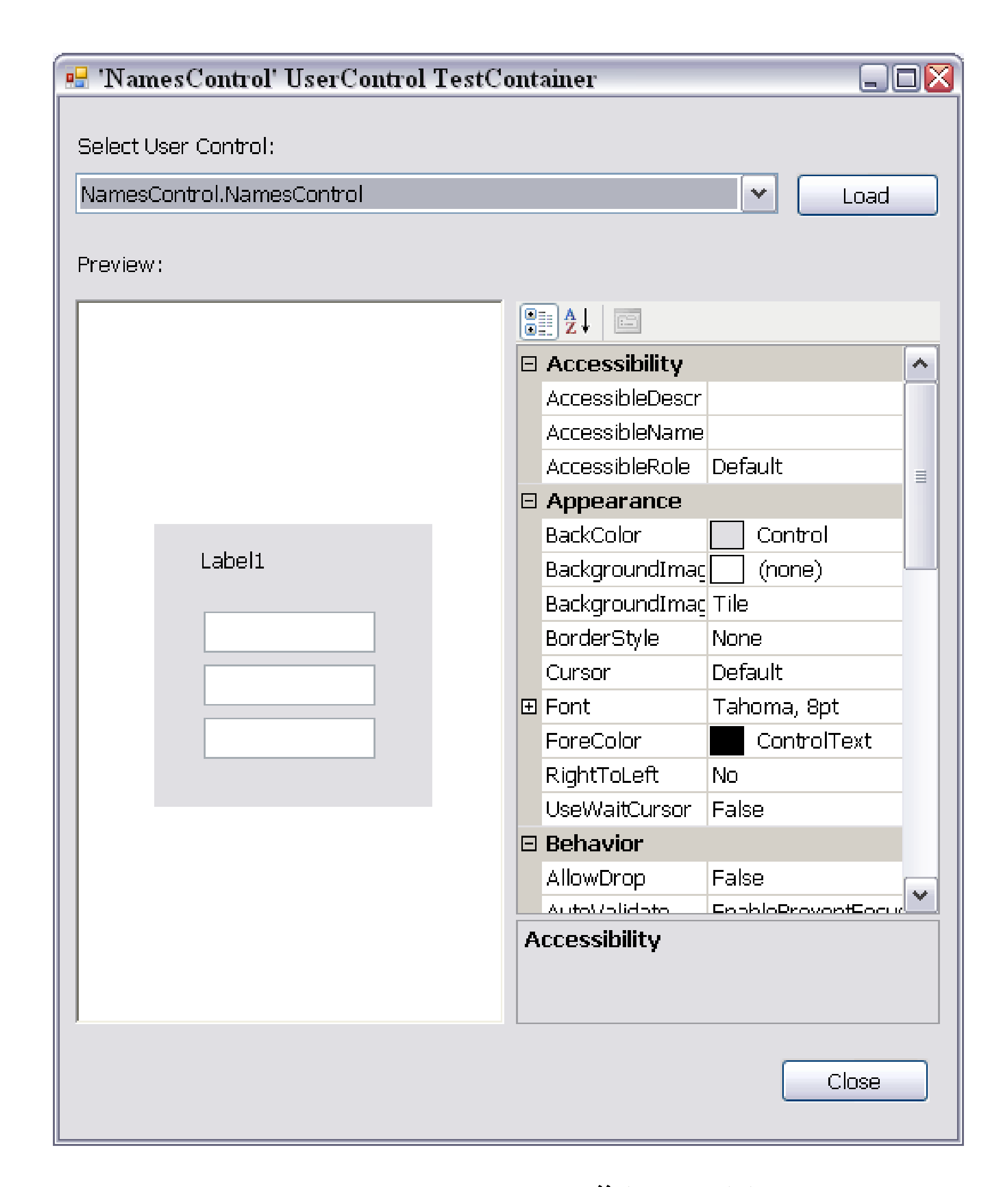
في النافذة Solution Explorer اختار NamesControl.vb ثم من القائمة View اختار Code.

في محرر التعليمات أ ضف التعليمات التالـ ية للحدث FirstName_TextChanged.

```
Private Sub FirstName_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
FirstName.TextChanged, MiddleName.TextChanged,
LastName.TextChanged
   ' Display the contents of the three text boxes in
the label.
   FullName.Text = FirstName.Text & " " &
MiddleName.Text & " " & LastName.Text
End Sub
```

من القائمة File اختار Save All.

أضرب المفتاح F5 لاختبار البرنا مج. تظ هر النافذة User Control Test Container المبينة في شكل 77-10. قم باختبار العنصر في النافذة Preview.



شكل 10-77: النافذة User Control Test Container

أنقر فوق المفتاح Close لإنهاء اختبار عنصر التحكم.

1.64.2خصائص عنصر التحكم

الخصـائص بالنسبة لعناصر التحكم القياسية تسمح للمبرمج أن يقوم بضبط والتعرف على قيم محددة حول عنصر التحكم في أثناء تصميم وتشغيل التطبيق. وبالمثل يرغب مصمم عنصر التحكم في أن يكون لعنصر تحكمه خصائص يمكن للمطورين الذين سوف يستخدمون هذا العنصر أن يتحكموا فيها، وتظهر في نافذة Properties كأي عنصر تحكم قياسي.

تصميم الخصائص لعناصر التحكم يشبه إلى حد كبير تصميم خصائص الفئات، لكنه يختلف عنه في إنه يمكنك أن استخدام خصائص عناصر التحكم التي يتكون منها عنصر تحكمك. كما في الفئات يمكنك أن تعلن عن الخصائص ثم تضيف التعليمات للإجراءات Get و Set.

كما هو واضح الآن، ليس هناك ثمة طريقة لاسترجاع القيم التي أدخلت إلى عناصر التحكم FirstName و MiddleName و LastName ولا تلك التي ظهرت في FullName، حيث يجب على المبرمج أن يصمم خصائص تتيح لمستخدمي عنصر التحكم خاصته أن يتعاملوا مع هذه القيم بالتخصيص أو بالاسترجاع.

لتصميم خصائص عنصر التحكم الذي ننشئه سوف نقوم بإجراء الخطوات التالية:

273.أفتح المشروع NamesUserControl.

في النافذة Solution Explorer اختار NamesControl.vb ثم من القائمة View اختار Code.

في محرر التعليمات أضف التعليمات التالية عقب Public Class مباشرة.

```
Property FirstNameText() As String
  Get
    Return FirstName.Text
  End Get
  Set(ByVal value As String)
    FirstName.Text = value
  End Set
End Property
Property MiddleNameText() As String
  Get
    Return MiddleName.Text
  End Get
  Set(ByVal value As String)
    MiddleName.Text = value
  End Set
End Property
```

```
Property LastNameText() As String
   Get
      Return LastName.Text
   End Get
   Set(ByVal value As String)
      LastName.Text = value
   End Set
End Property
```

من القائمة File أختر Save All.

أضرب المفتاح F5 لتشغيل البرنامج. انتقل إلى أسفل النافذة Properties في النافذة User Control Test Container لتجد الخصائص الثلاثة، اكتب في القيمة FirstNameText لتجد هذه القيمة تتغير في عنصر التحكم وبالتالي في عنصر التحكم FullName.

AutoScroll	False
⊞ AutoScrollMargi	0, 0
⊞ AutoScrollMinSi	0, 0
AutoSize	False
AutoSizeMode	GrowOnly
Dock	None
⊞ Location	**0, 0**
⊞ Margin	3, 3, 3, 3
⊞ MaximumSize	0, 0
⊞ MinimumSize	0, 0
⊞ Padding	0, 0, 0, 0
⊞ Size	**162, 165**
⊟ **Misc**	
FirstNameText	
LastNameText	
MiddleNameTex	

شكل 10-78: الخصائص الجديدة تظهر في نهاية نافذة Properties.

1.64.3 القيم المسماة

كل الخصائص التي أضفناها إلى عنصر التحكم الذي أنشأنه حتى الآن تعتد على قيم نصية، لكن الخصائص يمكن ان تتعامل مع أنواع بيانات مختلفة. احيانا يقوم المبرمج بعمل قائمة من القيم لإختيار منها، مثل الخاصية SizeMode لعنصر التحكم PictureBox.

لنفترض أننا نريد عمل خاصية تتيح لنا أن نتحكم في الهيئة التي سوف يبدو عليها النص المعروض في عنصر التحكم FullName. في هذه الحالة يمكننا أن نضيف قائمة لهذه الخاصية تضم First Name First و Last Name First و هكذا.

تسمح Visual Basic أن نستخدم الخاصية enumeration الذي يضم القيم التي تريد ها. الخاصية enumeration هي عبارة عن قائمة رقمية ويخصص المبرمج مسمى لكل قيمة رقمية فيها، ويقوم بإعلان هذه الخاصية مستخدما الكلمة المحجوزة Enum كما في المثال التالي:

```
Public Enum Display
   FirstMiddleLast
   FirstLast
   LastFirstMiddle
   LastFirst
End Enum
```

وبمجرد عمل الـ enumeration يمكن استخدامه كأي من أنواع البيانات المعتادة. ولإضافة الخاصية التي تعرض قائمة القيم، يجب أن تقوم في البداية متغير من ذات الفئة كـ Enum ثم تعلن الخاصية من ذات النوع. أثناء تصميم التطبيق تظهر قائمة القيم في نافذة Properties.

لنقوم بتجربة هذه الخاصية مع عنصر التحكم الذي قمنا بإنشائه.

274.أفتح المشروع NamesUserControl.

في النافذة Solution Explorer اختار NamesControl.vb ثم من القائمة View اختار Code.

في محرر التعليمات أضف التعليمات التالية عقب الخصائص.

```
Public Enum Display
   FirstMiddleLast
   FirstLast
   LastFirstMiddle
   LastFirst
End Enum
```

أضف التعليمات التالية لإضافة خاصية جديدة.

```
Private DisplayStyleList As Display
Property DisplayStyle() As Display
   Get
      Return DisplayStyleList
   End Get
   Set(ByVal value As Display)
      DisplayStyleList = value
   End Set
End Property
```

أحذف التعليمات الموجودة في الحدث FirstName_TextChanged واستبدلها بما يلي:

```
Select Case DisplayStyleList
 Case Display.FirstLast
  FullName.Text = FirstName.Text & " " & LastName.Text
 Case Display.FirstMiddleLast
  FullName.Text = FirstName.Text & " " &
MiddleName.Text & " " & LastName.Text
 Case Display.LastFirst
  FullName.Text = LastName.Text & ", " & FirstName.Text
 Case Display.LastFirstMiddle
  FullName.Text = LastName.Text & ", " & FirstName.Text
& " " & MiddleName.Text
End Select
```

من القائمة File أختر Save All.

أضرب المفتاح F5 لتشغيل البرنامج. أنتقل إلى نهاية النافذة Properties وقم بتغيير الخاصية DisplayStyle أكثر من مرة وفي كل مرة جرب إدخال القيم الثلاثة للاسم.

1.65 اختبار عنصر التحكم

بمجرد الانتهاء من تصميم عنصر التحكم واختباره في النافذة User Control Test Container يمكنك – بل ينصح بذلك – أن تقوم باختباره في تطبيق مبسط، يمكنك أن تفعل ذلك ببساطة عن طريق إضافة مشروع Windows Application لمشروعك الأساسي. وعندئذ يظهر عنصر التحكم الذي قمت بإنشائه آليا في صندوق الأدوات بحيث يمكنك سحبه ورميه على نموذج مشروعك.

لنقوم بهذا سوف نتبع الإجراءات التالية:

275.أفتح المشروع NamesUserControl.

276.من النافذة File اختر Add ومنها New Project.

277.من النافذة Add New Project أختار Windows Application.

278.في الخانة Name أكتب اسم المشروع UserControlTest ثم أنقر Ok.

يظهر مشروع جديد في النافذة Solution Explorer.

279. من النافذة Solution Explorer اختار المشروع UserControlTest ثم من القائمة Project اختار Set as StartUp Project.

280. من صندوق الأدوات قم بسحب عنصر لتحكم NamesControl وارميه فوق النموذج.

281.قم بضبط الخصائص FirstName و MiddleName و LastName في النافذة Properties.

282.من القائمة File أختار Save All.

283.أضرب المفتاح F5 في لوحة المفاتيح لاختبار البرنامج.

1.66تحسين عنصر التحكم المنشئ

فيما سبق قمنا بإضافة عنصر التحكم الذي قمنا بإنشائه على مشروع واختبرناه وتأكدنا من انه يعمل كما يجب. هناك بعض التحسينات التي ربما تكون قد شعرت انه يجب

أضافتها على عنصر التحكم الذي قمنا بإنشائه، فمثلا ربما تحتاج أن تتأكد أن الأسماء الثلاثة قد تم تخصيصهم.

وحتى يكون عنصر التحكم مفهوم ربما نضيف مسميات لكل TextBox، لكن ماذا إذا رغب المبرمج في أن يستبدل الاسم الأوسط بحرف فقط. ربما يون الأمثل في هذه الحالة هو ان نقوم بعمل خاصية يمكن التحكم فيها أثناء التصميم.

1.66.1تحسين مظهر عنصر التحكم

سوف نتبع الإجراءات التالية لتحسين مظهر عنصر التحكم الذي قمنا بإنشائه.

284.أفتح المشروع NamesUserControl.

في النافذة Solution Explorer اختار NamesControl.vb ثم من القائمة View اختار Designer.

قم بسحب عنصر تحكم Label من صندوق الأدوات وارميه فوق قالب التصميم امام عنصر التحكم FirstName وكرر هذا العمل لكل من عنصري التحكم MiddleName و LastName.

في النافذة Solution Explorer اختار NamesControl.vb ثم من القائمة View اختار Code.

أضف التعليمات التالية في نافذة المحرر لعمل خصائص لعناصر التحكم التي اضفناها:

```
Private text1 As String = "First Name"
Property Label1Text() As String
  Get
    Return text1
  End Get
  Set(ByVal value As String)
    text1 = value
    Label1.Text = text1
  End Set
```

```
End Property
Private text2 As String = "Middle Name"
Property Label2Text() As String
   Get
      Return text2
   End Get
   Set(ByVal value As String)
      text2 = value
      Label2.Text = text2
   End Set
End Property
Private text3 As String = "Last Name"
Property Label3Text() As String
   Get
      Return text3
   End Get
   Set(ByVal value As String)
      text3 = value
      Label3.Text = text3
   End Set
End Property
```

لا حظ أن هذه التعليمات تعرف خصائص الـ Labels من النوع Private بينما الإعلان عن الخاصية يتضمن القيمة الافتراضية لها.

من نافذة محرر التعليمات، اختر NamesControl Events من قائمة الكائنات الموجود على اليسار فوق، و من قائمة الأحداث الموجودة فوق على اليمين اختار الحدث Load.

أضف التعليمات التالية للحدث NamesControl_Load:

```
' Initialize the three labels
Me.Label1.Text = Label1Text
Me.Label2.Text = Label2Text
Me.Label3.Text = Label3Text
```

من القامة Build اختار Build Solution.

من النافذة Solution Explorer اختار النموذج Form1 ثم من القائمة View اختار Designer. حاول ان تقوم بتغيير الخصائص الخاصة بعنصر التحكم.

من القائمة File اختار Sava All.

1.66.2الحدث Validating

من المفاهيم الهامة التي يجب أن نتعرض لها عند تصميم عنصر تحكم هو اختبار ما إذا كان القيم المدخلة له – أو القيم التي تضبط خصائصه – ملائمة لما عليه التصميم. معظم عناصر التحكم لها الحدث Validating والذي يعمل عندما ينتقل التحكم من عنصر التحكم إلى عنصر تحكم آخر. في هذا الحدث يمكن أن تضيف التعليمات التي سوف تختبر ما إذا كان كل textbox يحتوي على قيمة حرفية أم لا. فإذا كان أحد هذه العناصر فارغاً تخرج رسالة تنبه المستخدم ان هذه الصندوق فارغ. ويمكن أن يستمد نص هذه الرسالة من خاصية يقوم مستخدم العنصر بضبطها كيفما شاء. كما انه من الجائز أن مستخدم برنامجك لا يستخدم الاسم الأوسط، لذلك يمكننا إضافة خاصية من النوع الـ Boolean لتوقف عمل حدث التحقق على الـ MiddleName.

لنقوم بهذا لنتبع الخطوات التالية:

285.في محرر التعليمات أضف تعليمات لخاصيتين سوف يتم التحقق منهما، الأولى لتبين ما إذا كان الاسم الأوسط مطلوب أم لا. والثانية تبين الرسالة التي سوف يتم إرسالها. وذلك بإضافة التعليمات التالية:

```
Private required As Boolean = True
Property MiddleNameRequired() As Boolean
   Get
      Return required
   End Get
   Set(ByVal value As Boolean)
      required = value
   End Set
End Property
```

```
Private errormessage As String = "Please enter your name."
Property ValidationErrorMessage() As String
   Get
      Return errormessage
   End Get
   Set(ByVal value As String)
      errormessage = value
   End Set
End Property
```

في محرر التعليمات، اختار namesControl Events من قائمة الكائنات والحدث Validating من قائمة الأحداث.

اصف التعليمات التالية للحدث NamesControl_Validating:

```
If MiddleNameRequired = True Then
   If FirstName.Text = "" Or MiddleName.Text = "" Or _
LastName.Text = "" Then
      MsgBox(ValidationErrorMessage)
   End If
Else
   ' Middle name isn't required.
   If FirstName.Text = "" Or LastName.Text = "" Then
      MsgBox(ValidationErrorMessage)
   End If
End If
```

من القائمة Build اختار Build Solution.

من النافذة Solution Explorer اختار النموذج Form1 ثم من القائمة View اختار Designer. حاول ان تقوم بتغيير الخصائص الخاصة بعنصر التحكم.

قم بسحب عنصر تحكم Button من صندوق الأدوات وارميه فوق النموذج Form1.

من القائمة File اختار Sava All.

اضرب المفتاح F5 ثم قم باختبار البرنامج.

برمجة الرسوم

توفر Visual Basic.NET 2008 مجموعة من الأدوات التي تساعد المبرمج على أداء الوظائف المتعلقة بالرسوم. هذه الأدوات هي موضوع هذا القسم.

1.67أظهار الرسوم

في قسم سابق تعلمت استخدام عنصر التحكم PictureBox لإظهار صورة فوق النموذج، هذه الطريقة مفيدة فقد إذا كانت الصورة المطلوب عرضها متوفرة، لكن في بعض الأحيان يحتاج الـ مبرمج أن يقوم بإضا فة ر سوم إلى نموذ جه لأهداف أخرى غ ير عرض الصور، كأن يضيف دائرة حمراء حول صندوق نصوص مهم مثلا. لأداء مثل هذه الوظائف فإن Visual Basic.NET 2008 توفر الفئة Graphics التي تتيح للمبرمج تنفيذ مختلف العمليات المتعلقة بالرسوم.

1.67.1أسس الرسم

قبل البدء في التعامل مع الرسوم من خلال Visual Basic.NET 2008 من الأفضل ان نتعرف على بعض المفاهيم الأساسية. تتكون شاشة الحاسوب من الآلاف من النقاط الصغيرة التي يطلق عليها الاسم بكسلات Pixels، بتخصيص لون لكل بكسل من هذه البكسلات تظهر الأشكال المختلفة فوق الشاشة.

الآن لنتخيل النموذج كلوحة رسم Canava سوف تقوم بالرسم فوقها كيفما شئت كما لوحة الرسم الحقيقية. وكما أن لوحة الرسم لها أبعاد تقاس بالبوصة أو بالسم، كذلك فإن للنموذج أبعاد تقاس بالبكسل. يستخدم في تحديد موقع البكسل فوق النموذج نظام إحداثي بسيط يلعب يمتد المحور السيني فيه من اليسار على اليمين، بينما يمتد المحور الصادي من الأعلى على الأسفل.

في هذا النظام الإحداثي تعتبر نقطة الأصل هي الركن العلوي الأيسر من الشاشة، لذلك فإنه إذا أردت رسم نقطة تبعد عشرة بكسلات من اليسار وعشرة بكسلات من أعلى، فإنك تشير إلى إحداثها السيني والصادي بالقيمتين ١٠ و ١٠.

كذلك تستخدم البكسلات لتوضيح طول وعرض الرسومات، فمثلا لرسم مربع طول حرفه ١٠٠ بكسل، بحيث يكون ركنه العلوي الأيسر عند النطة ١٠ و ١٠ نحتاج أن نبين أن أبعاده هي ١٠ و ١٠ و ١٠٠ و ١٠٠ حيث ١٠ و ١٠ هي نقطة الركن العلوي الأيسر، ١٠٠ طوله، ١٠٠ الثانية عرضه.

عملية الرسم فوق النموذج يطلق عليها الاسم painting، النماذج وعناصر التحكم لها حدث يطلق عليه Paint والذي يحدث عندما تحتاج هذه النماذج وعناصر التحكم إلى إعادة رسمها، على سبيل المثال، عند ما يعرض نموذج للمرة الأولى، أو يختد في خلف نفاذة أخرى، فإن أي تعليمات قمت بكتابتها لعرض رسوم فوق هذا النموذج أو عنصر التحكم يتم اختزانها في الحدث Paint.

1.67.2رسم خط

لرسم خط عبر النموذج، فإن هناك شيئين سيكون عليك تعريفهما، إحداثياته ولونه. وكما بينا سلفاً فإن الإحداثيات يتم تعيينها باستخدام البكسلات. بالنسبة للخط هناك زوج من الإحداثيات التي تحتاجها، إحداثيات نقطة البداية وإحداثيات نقطة النهاية.

وكما أنك تحتاج إلى قلم عندما رسم خط فوق لوحة الرسم، كذلك فإنك تحتاج إلى مثل هذا الكائن عند رسم الخط فوق النموذج. ويطلق على مثل هذا الكائن لحسمن الحظ الاسم Pen. الكائن Pen يعين الهيئة التي سوف يظهر بها الخط، وفي معظم الحالات تكوه هذه الهيئة قاصرة على لون الخط. في التدريب التالي سوف نقوم برسم خطوط أفقية ورأسية وعرضية فوق النموذج باستخدام تعليمات Visual Basic.NET 2008.

286.أختار New Project من القائمة File.

من التبويب Templates اختار Windows Application.

في الخانة Name أجعل اسم المشروع Lines ثم أنقر المفتاح Ok.

أنقر مرتين فوق النموذج لتظهر نافذة تحرير التعليمات، أختار الحدث Paint من قائمة الأحداث.

أضف التعليمات التالية للحدث Form1_Paint:

```
' Draw a 400 pixel black line 25 pixels from the top of the form.
e.Graphics.DrawLine(Pens.Black, 0, 25, 400, 25)
' Draw a 500 pixel red line 100 pixels from the left of the form.
e.Graphics.DrawLine(Pens.Red, 100, 0, 100, 500)
' Draw a diagonal blue line from the upper left to the lower right.
e.Graphics.DrawLine(Pens.Blue, 0, 0, Me.Width, Me.Height)
```

أنقر المفتاح F5 لتنفيذ البرنامج.

1.68رسم الأشكال

فيما سبق تعرفنا كيف نرسم خط بسيط فوق النموذج باستخدام الطريقة DrawLine و كائن Pen، إضافة إلى رسم الخطوط يمكن استخدام Visual Basic.NET 2008 لرسم الإشكال سواء كانت الفارغة أو المصمتة، لكن عند رسم الأشكال المصمتة نحتاج إلى كائن جديد – شبيه بالكائن Pen – هو الكائن Brush.

1.68.1رسم الأشكال البسيطة

رسم الأشكال يتشابه مع رسم الخطوط حيث تكون مضطر لتعيين إحداثيات أشكالك وهيئتها. وعلى حين يحتاج رسم الخط تحديد إحداثيات نقطتين فحسب، فإن الأشكال مثل المربع أو المستطيل تحتاج إحداثيات تبين الركن العلوي الأيسر منها إضافة إلى طولها وعرضها.

الدائرة والشكل البيضاوي لا تحتاج في رسمها الركن الأعلى الأيسر منها – حيث لا يوجد لها مثل هذه النقطة – لكن عوضاً عن هذا يستخدم لتحديدها الركن العلوي الأيسر للمستطيل الوهمي المحيط بها.

فيما يلي تدريب على رسم شكل بسيط فوق النموذج.

287.أختار New Project من القائمة File.

من التبويب Templates اختار Windows Application.

في الخانة Name أجعل اسم المشروع Shapes ثم أنقر المفتاح Ok.

أنقر مرتين فوق النموذج لتظهر نافذة تحرير التعليمات، أختار الحدث Paint من قائمة الأحداث.

أضف التعليمات التالية للحدث Form1_Paint:

```
' Draw a 200 by 150 pixel green rectangle.
e.Graphics.DrawRectangle(Pens.Green, 10, 10, 200, 150)
' Draw a blue square
e.Graphics.DrawRectangle(Pens.Blue, 30, 30, 150, 150)
' Draw a 150 pixel diameter red circle.
e.Graphics.DrawEllipse(Pens.Red, 0, 0, 150, 150)
' Draw a 250 by 125 pixel yellow oval.
e.Graphics.DrawEllipse(Pens.Yellow, 20, 20, 250, 125)
```

أنقر المفتاح F5 لتنفيذ البرنامج.

1.68.2رسم الأشكال المصمتة

الأشكال الـتي قمـنا برسمها في التدريب السـابق كـانت تعتمد على الـخط الخارجي Outline فقط، لرسم أشكال مصمتة يحتاج المبرمج استخدام طرق تعبئة اللون مثل FillRectangle و FillEllipse. هذه الطرق تستخدم الكائن Brush و هو كائن رسومي آخر.

عندما تقوم بتعبئة الشكل بلون مختلف، ستكون محتاج إحداثيات مثل التي استخدمتها لرسم الشكل، وإلا سوف تجد الخط الخارجي قد اختفى تحت اللون الجديد. فمثلاً، لتعبئة المربع ذي الإحداثيات 0,0,150,150 يجب تعيين إحداثيات التعبئة لتكون 1,1,148,148 وذلك على أساس أن سمك الخط الخارجي بكسل واحد.

لنجرب ذلك اتبع الخطوات التالية:

288.أضف التعليمات التالية للحدث Form1_Paint:

```
' Fill the circle with the same color as its border.
e.Graphics.FillEllipse(Brushes.Red, 0, 0, 150, 150)
' Fill the square with a different color.
e.Graphics.FillRectangle(Brushes.Aquamarine, 31, 31,
148, 148)
```

أنقر المفتاح F5 لتنفيذ البرنامج.

1.69رسم النصوص فوق النموذج

سوف نتعلم فيما يلي كيف نقوم برسم النصوص فوق النموذج مستخدمين طرق الرسوم Graphics Methods. في أقسام سابقة تعلمنا كيف نقوم باستخدام عنصر التحكم Label لعرض النصوص. لكن هناك حالات يحتاج فيها المبرمج أن يضيف النصوص فيها إلى برنامجه كرسوم، فمثلا يمكن أن نقدم الخط مائل باستخدام عنصر التحكم Label لكن بتقديمه كرسوم نستطيع أن نميل النص إلى أي اتجاه كيفما شئنا.

1.69.1رسم النص

لرسم النص فوق النموذج سوف نستخدم الطريقة DrawString. وكما في طرق الرسم الأخرى، تعتمد الطريقة DrawString على كائن Brush لتحديد لون وإحداثيات النص المرسوم. وتمثل نقطة الركن العلوي الأيسر هي نقطة ركن المستطيل المحيط بالنص.

تحتاج الطريقة DrawString أيضاً إلى قيمتين أخريين، قيمة النص المطلوب رسمه، ونوع الخط المستخدم لرسم النص. ولتمثيل الخط لا بد من إنشاء كائن Font ثم استخدام هذا الكائن في الطريقة DrawString.

سوف نتبع الخطوات التالية لاختبار طريقة رسم النص.

289.أختار New Project من القائمة File.

من التبويب Templates اختار Windows Application.

في الخانة Name أجعل اسم المشروع DrawText ثم أنقر المفتاح Ok.

أنقر مرتين فوق النموذج لتظهر نافذة تحرير التعليمات، أختار الحدث Paint من قائمة الأحداث.

أضف التعليمات التالية للحدث Form1_Paint:

```
' Create a font object.
Dim aFont As New System.Drawing.Font("Arial", 22,
FontStyle.Bold)
' Display the text with the DrawString method.
```

```
e.Graphics.DrawString("Graphics are fun!", aFont,
Brushes.Black, 20, 10)
```

أنقر المفتاح F5 لتنفيذ البرنامج.

1.69.2المؤثرات على النصوص

لرسم نص مائل أو مدور نحتاج طريقة أخرى هي الطريقة transform. هناك أنواع مختلفة من الـ transform الممكن استخدامها للحصول على مؤثرات مختلفة للنصوص، لكننا سوف نقتصر على طريقة واحدة هي RotateTransform.

هذه الطريقة تحتاج إلى بارمتر واحد يمثل الزاوية التي سوف يدروها النص. عملية التحويل تؤثر على السطر التالي للسطر الذي يحتوي على تعليمات الـ Transform. هذه الطريقة يمكن أن تنسحب أيضاً على الأشكال والخطوط.

لنجرب هذا سوف نتبع الخطوات التالية:

290.أضف التعليمات التالية للحدث Form1_Paint:

```
' Fill the circle with the same color as its border.
Dim aFont As New System.Drawing.Font("Arial", 22,
FontStyle.Bold)
e.Graphics.FillEllipse(Brushes.Red, 0, 0, 150, 150)
' Fill the square with a different color.
e.Graphics.FillRectangle(Brushes.Aquamarine, 31, 31,
148, 148)
```

أنقر المفتاح F5 لتنفيذ البرنامج.

1.70رسم الصور

فيما يلي نتعلم كيف يمكننا أن نقوم بإضافة صور باستخدام طرق الفئة graphics.

في قسم مبكر، قمنا بعرض الصور مستخدمين عنصر التحكم PictureBox. تتيح الفئة graphics للمبرمج قراءة صورة من ملف.

لنقوم بهذا سوف نتبع الخطوات التالية:

291.أختار New Project من القائمة File.

من التبويب Templates اختار Windows Application.

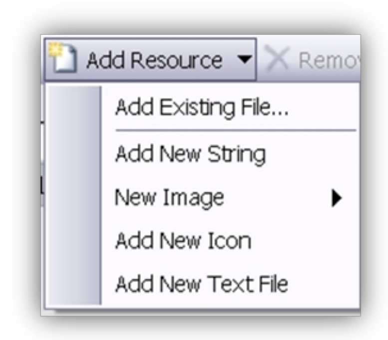
في الخانة Name أجعل اسم المشروع DrawImage ثم أنقر المفتاح Ok.

في النافذة Solution Explorer نقر مستخدما مفتاح الفارة اليمن فوق المشروع DrawImage واختار Properties.

أختار التبويب Resources.

أنقر فوق السهم المجاور لت Add Resources واختار من القائمة المنسدلة Add Existing File – أنظر شكل 11-79.

استخدم نافذة اختيار الملفات لتختار ملف صورة.



شكل 11-79: أختار Add existing File

أنقر مرتين فوق النموذج لتظهر نافذة تحرير التعليمات، أختار الحدث Paint من قائمة الأحداث.

أضف التعليمات التالية للحدث Form1_Paint:

```
e.Graphics.RotateTransform(15)
e.Graphics.DrawImage(My.Resources.picture, 150, 50)
```

استبدل picture باسم الصورة التي استخدمتها.

أنقر المفتاح F5 لتنفيذ البرنامج.



شكل 11-80: الصورة كما ظهرت التطبيق